EUROCG'25

Liblice, April 9-11 2025

BOOKLET OF ABSTRACTS

Preface

The 41st European Workshop on Computational Geometry (EuroCG 2025) was held on April 09-11, 2025 in Liblice, Czech Republic. EuroCG is an annual, informal workshop whose goal is to provide a forum for scientists to meet, present their work, interact, and establish collaborations, in order to promote research in the field of Computational Geometry, within Europe and beyond.

Concerning the scientific program, we received 88 submissions, which underwent a refereeing process by the program committee. We selected 81 submissions for presentation at the workshop; afterwards, 1 submission was withdrawn. EuroCG does not have formally published proceedings; therefore, we expect most of the results outlined here to be also submitted to peer-reviewed conferences and/or journals. This book of short abstracts, available through the EuroCG 2025 website, should be regarded as a collection of preprints. In addition to the 80 contributed talks, this book contains abstracts of the invited lectures. The invited speakers were Oswin Aichholzer (Graz University of Technology, Austria), Eva Rotenberg (Technical University of Denmark), and Uli Wagner (Institute of Science and Technology Austria). For the workshop, we had 128 registered participants.

Many thanks to all authors and to the members of the program committee and all external reviewers for their insightful comments. We also thank the organizing committee members: Martin Balko (Charles University), Pavel Hubáček (Czech Academy of Sciences), Anna Kotešovcová (Conforg), Maria Saumell (Czech Technical University in Prague), and Pavel Valtr (Charles University). Finally, we are very grateful for the generous support of our Gold Sponsors, RSJ and Unicorn, and to our Sponsor, Znovín, and to our Contributor, Department of Theoretical Computer Science, Faculty of Information Technology CTU in Prague.

The conference gave out a best student presentation award. The prize was voted by the EuroCG 2025 attendees to recognize the effort of young researchers to present their work clearly and elegantly. The winner was Johanna Ockenfels, for the presentation of the paper "Chasing puppies on orthogonal straight-line plane graphs". Congratulations to Johanna!

During the business meeting, Alexandra Weinberger presented the 2026 edition of EuroCG, which will take place in Hagen, Germany. A single bid was presented for 2027 by Christiane Schmidt and, as a consequence, EuroCG 2027 will take place in Norrköping, Sweden.

Looking forward to seeing you next year all in Hagen.

April 2025
Jan Kratochvil and Giuseppe Liotta

Program Committee

Michael A. Bekos	University of Ioannina
Carla Binucci	University of Perugia
Sergio Cabello	University of Ljubljana
Jean Cardinal	Université Libre de Bruxelles
Éric Colin de Verdière	CNRS, LIGM, Marne-la-Vallee
Sabine Cornelsen	University of Konstanz
Sándor Fekete	TU Braunschweig
Stefan Felsner	TU Berlin
Petr Hliněný	Masaryk University
Michael Hoffmann	ETH Zürich
Fabian Klute	UPC Barcelona
Jan Kratochvíl (co-chair)	Charles University
Giuseppe Liotta (co-chair)	University of Perugia
Rahnuma Islam Nishat	Brock University
Yoshio Okamoto	University of Electro-Communications
Dömötör Pálvölgyi	Eötvös Loránd University
Charis Papadopoulos	University of Ioannina
Zuzana Patáková	Charles University
Joseph O'Rourke	Smith College
Paweł Rżązewski	Warsaw University of Technology and University of Warsaw
Gelasio Salazar	Universidad Autónoma de San Luis Potosí
Christiane Schmidt	Linköping University
Miloš Stojaković	University of Novi Sad
Alessandra Tappini	University of Perugia
Pavel Valtr	Charles University
Marc van Kreveld	Utrecht University
Birgit Vogtenhuber	Graz University of Technology
Meirav Zehavi	Ben-Gurion University
Johannes Zink	Technische Universität München

Organizing Committee

Martin Balko (co-chair)	Charles University
Pavel Hubáček	Czech Academy of Sciences
Anna Kotešovcová	Conforg
Maria Saumell (co-chair)	Czech Technical University
Pavel Valtr	Charles University

External Reviewers

Péter Ágoston
Yan Alves Radtke
Jakub Balabán
Aaron Becker
Jesse Beisegel
Therese Biedl
Annika Bonerath
Timo Brand
Anna Brötzner
Jacobus Conradi
Daniel Cranston
Arun Kumar Das
Alexander Dobler
Ruy Fabila-Monroy
Omrit Filtser
Krzysztof Fleszar
David Flores
Jonas Friemel
Henry Förster
Florian Galliot
Loukas Georgiadis
Panos Giannopoulos
Valentin Gledel
Luca Grilli
Fabrizio Grosso
Tim Hegemann
César Hernández Vélez
Jan Jedelský
Marija Jelić Milutinović
Attila Jung
Eleni Katsanou
Vahideh Keikha
Balázs Keszegh
Boris Klemz
Bettina Klinz
Katharina Klost
Athanasios Konstantinidis
Ramin Kosfeld
Peter Kramer
Dominik Krupke
Csenge Lili Ködmön
Robert Lauff
Jesus Leanos
Chek-Manh Loi
Bradley McCoy
Wolfgang Mulzer
Bengt J. Nilsson
Tim Ophelders
Giacomo Ortali
Irene Parada
Maria Eleni Pavlidi
Michael Perk
Daniel Perz
Tom Peters
Maximilian Pfister
Tommaso Piselli
Valentin Polishchuk
Marko Radovanovic
Carolin Rehs
Christian Rieck
Bastien Rivier
Sandro Roch
Edgardo Roldán-Pensado
Maria Saumell
Marko Savić
Patrick Schnider
Raimund Seidel
Marie Diana Sieper
Rodrigo Silveira
Anna Slivková
Frank Staals
Sabine Storandt
Antonios Symvonis
Viet Cuong Than
Csaba Toth
Géza Tóth
Giorgio Vinciguerra
Mirko H. Wagner
Tobias Wallner
Alexandra Weinberger
Alexandra Wesolek
Mathijs Wintraecken
Samuel Wolf
Alexander Wolff
Kristóf Zólomy

Table of Contents

Flips in Plane Graphs - Old Problems, New Results

Oswin Aichholzer[1]

1 Graz University of Technology
`oswin.aichholzer@tugraz.at`

Abstract

Reconfiguration is the process of changing a structure into another - either through continuous motion or through discrete changes. We concentrate on plane graphs and discrete reconfiguration steps of bounded complexity, like exchanging one or two edge(s) of the graph for one or two other edge(s), which is often called a flip. The flip graph is defined as the graph having a vertex for each configuration (in our case for each plane graph of a certain type) and an edge for each flip between them. Three questions are central: studying the connectivity of the flip graph, its diameter, and the complexity of finding the shortest flip sequence between two given configurations. Many classic and new results are known, for example for flips in triangulations or the transformation of plane spanning trees. We will give an overview of these results and mention several (old and new) open problems in this area, accompanied by recent developments for matchings, trees, paths, and triangulations.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

How to Draw a Changing Graph

Eva Rotenberg[1]

1 Technical University of Denmark
`erot@dtu.dk`

Abstract

A graph is planar if it can be drawn in the plane without its edges crossing (except at the endpoints). The algorithmic question of whether a graph admits a planar embedding can be decided in linear time [Hopcroft,Tarjan '74]. In this talk, we study the dynamic setting. Given a graph, consider that edges are inserted and deleted by an adversary. Can we efficiently update the answer to the yes/no question of whether the graph admits a planar embedding? This talk presents a deterministic algorithm for updating the answer to said question; whether the graph is presently planar [Holm,R '20]. Our algorithm runs in an amortized time per update that is asymptotically cubic in the logarithm of the size of the graph; $O(\log^3 n)$. Curiously, our solution for maintaining that one bit of information goes via implicitly maintaining information about a feasible planar embedding of the present graph in the affirmative case (and a planar embedding of a planar subgraph, otherwise). A natural next question is to study the notion of upward planarity, a restricted planar embedding of a directed graph, in which every directed edge must be embedded as a y-monotone curve. Here, we present some partial results [van der Hoog,Parada,R '24] and an open problem.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.

Face Numbers of Polytopes and Levels in Arrangements

Uli Wagner[1]

1 **Institute of Science and Technology Austria**
uli@ist.ac.at

Abstract

Levels in arrangements (and the dual notion of k-sets) play a fundamental role in discrete and computational geometry and are a natural generalization of convex polytopes (which correspond to the 0-level). We will survey some classical results from the combinatorial theory of convex polytopes, including the Dehn–Sommerville Relations, McMullen's Upper Bound Theorem, and the g-Theorem, due to Stanley and Billera–Lee, which completely characterizes the face numbers of simple polytopes. It is natural to wonder whether and to what extent this theory can be generalized to face numbers of levels in arrangements. We will discuss some steps in this direction, including both classical and new results, as well as various conjectures and open problems.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Computing Oriented Spanners and their Dilation

Kevin Buchin[1], Antonia Kalb[*1], Anil Maheshwari[†2], Saeed Odak[3], Carolin Rehs[1], Michiel Smid[†2], and Sampson Wong[4]

1 **TU Dortmund University, Germany**
`kevin.buchin, antonia.kalb, carolin.rehs@tu-dortmund.de`
2 **Carleton University, Canada**
`anil, michiel@scs.carleton.ca`
3 **University of Ottawa, Canada**
`Saeed.Odak@gmail.com`
4 **University of Copenhagen, Denmark**
`sampson.wong123@gmail.com`

Abstract

Given a point set P in a metric space and a real number $t \geq 1$, an *oriented t-spanner* is an oriented graph $\overrightarrow{G} = (P, \overrightarrow{E})$, where for every pair of distinct points p and q in P, the shortest oriented closed walk in $\overrightarrow{G}$ that contains p and q is at most a factor t longer than the perimeter of the smallest triangle in P containing p and q. The *oriented dilation* of a graph $\overrightarrow{G}$ is the minimum t for which $\overrightarrow{G}$ is an oriented t-spanner.

For arbitrary point sets of size n in $\mathbb{R}^d$, where $d \geq 2$ is a constant, the only known oriented spanner construction is an oriented 2-spanner with $\binom{n}{2}$ edges. Moreover, there exists a set P of four points in the plane, for which the oriented dilation is larger than 1.46, for any oriented graph on P.

We present the first algorithm that computes a sparse oriented spanner whose oriented dilation is bounded by a constant. More specifically, for any set of n points in the Euclidean space $\mathbb{R}^d$, where d is a constant, we construct an oriented $(2+\varepsilon)$-spanner with $\mathcal{O}(n)$ edges in $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space. Our construction uses the well-separated pair decomposition and an algorithm that computes a $(1+\varepsilon)$-approximation of the minimum-perimeter triangle in P containing two given query points in $\mathcal{O}(\log n)$ time.

We further prove that even if the oriented graph is already given, computing its oriented dilation is APSP-hard for points in a general metric space. We complement this result with an algorithm that approximates the oriented dilation of a given graph in subcubic time for point sets in $\mathbb{R}^d$, where d is a constant.

Related Version Full Version arXiv:2412.08165 [cs.CG]

1 Introduction

While geometric spanners have been researched for decades (see [3, 14] for a survey), directed versions have only been considered more recently. This is surprising since, in many applications, edges may be directed or even oriented if two-way connections are not permitted. Oriented spanners were first proposed in ESA'23 [4] and have since been studied in [5] and [6].

Given a point set P in the Euclidean space and a parameter t, an *oriented t-spanner* $\overrightarrow{G} = (P, \overrightarrow{E})$ is an oriented graph such that for every pair p, q of distinct points in P, the shortest oriented closed walk in $\overrightarrow{G}$ that contains p and q is at most a factor t longer than their shortest cycle in the complete undirected graph on P.

[*] *Antonia Kalb*: This work was supported by a fellowship of the German Academic Exchange Service (DAAD).
[†] *Anil Maheshwari*, *Michiel Smid*: funded by the Natural Sciences and Engineering Research Council of Canada (NSERC).

Formally, the *oriented dilation* is $t = \max_{p,q \in P} \left\{ \operatorname{odil}(p,q) = \frac{|C_{\overrightarrow{G}}(p,q)|}{|\Delta^*(p,q)|} \right\}$, where $C_{\overrightarrow{G}}(p,q)$ denotes the *shortest closed walk* containing p and q in the oriented graph $\overrightarrow{G}$ and $\Delta^*(p,q)$ denotes the *minimum-perimeter triangle* in P. This is the triangle Δ_{pqx}, where $x = \arg\min\{(|px| + |qx|) \mid x \in P \setminus \{p,q\}\}$.

Recall that a *t-spanner* is an undirected graph G with vertex set P, in which for any two points p and q, there exists a path between p and q in G whose length is at most t times the distance $|pq|$. In contrast to that, in the oriented setting, the shortest closed walk in the graph is compared to the minimum-perimeter triangle. In an oriented graph, comparing only the shortest path to the Euclidean distance would lead to arbitrary high dilations even on point sets with three vertices (see Figure 1). Therefore, the undirected dilation would not tell us much about the quality of an oriented spanner.

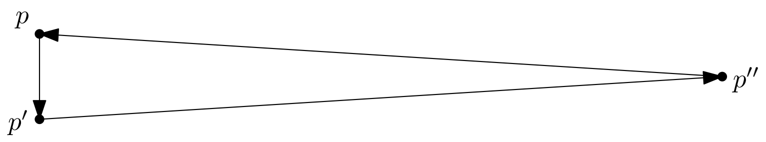


Figure 1 If p and p' are very close to each other and p'' is far away from both, any oriented graph will have arbitrarily high (directed) dilation [4].

Several algorithms are known that compute undirected $(1+\varepsilon)$-spanners with $\mathcal{O}(n)$ edges for any set of n points in $\mathbb{R}^d$. Examples are the greedy spanner [14], Θ-graphs [13] and spanners based on the well-separated pair decomposition (WSPD) [17]. Moreover, it is NP-hard to compute an undirected t-spanner with at most m edges [10].

In the oriented case, while it is NP-hard to compute an oriented t-spanner with at most m edges [4], the problem of constructing sparse oriented spanners has remained open until now. For arbitrary point sets of size n in $\mathbb{R}^d$, where $d \geq 2$ is a constant, the only known oriented spanner construction is a simple greedy algorithm that computes, in $\mathcal{O}(n^3)$ time, an oriented 2-spanner with $\binom{n}{2}$ edges [4]. If P consists of three vertices of an equilateral triangle in $\mathbb{R}^2$ and a fourth point in its centre, then the smallest oriented dilation for P is $2\sqrt{3} - 2 \approx 1.46$ [4] (see Figure 2). Thus, prior to our work, no algorithms were known that compute an oriented t-spanner with a subquadratic number of edges for any constant t.

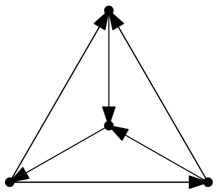


Figure 2 An oriented graph on 4 points with dilation $2\sqrt{3} - 2 \approx 1.46$ [4].

In this paper, we introduce the first algorithm to construct sparse oriented spanners for general point sets. For any set P of n points in $\mathbb{R}^d$, where d is a constant, the algorithm computes an oriented $(2+\varepsilon)$-spanner with $\mathcal{O}(n)$ edges in $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space.

While our approach uses the WSPD [7, 17], in contrast to undirected spanners, this does not immediately yield an oriented spanner. An undirected spanner can be obtained from a WSPD by adding one edge per well-separated pair. This does not work for constructing an oriented spanner because a path in both directions needs to be considered for every well-separated pair. To construct such paths, we present a data structure based on approximate nearest neighbour queries [2], that, given a pair of points, returns a point with an approximate minimum summed distance to the pair of the points, i.e. the minimum-perimeter triangle.

For the problem of computing the oriented dilation of a given graph, there is a straightforward cubic time algorithm. We show a subcubic approximation algorithm.

The hardness of computing the dilation of a given graph, especially in the undirected case, is a long-standing open question, and only cubic time algorithms are known [12]. It is conjectured that computing the dilation is as hard as the all-pairs shortest paths problem (APSP). In this paper, we demonstrate that computing the oriented dilation is APSP-hard.

2 Sparse Oriented Constant Dilation Spanner

We present the first construction for a sparse oriented $(2+\varepsilon)$-spanner for multidimensional point sets. Our algorithm consists of the following steps (a detailed version of Algorithm 1 is given in the full version of this paper):

Algorithm 1 $(2+\varepsilon)$-Spanner Construction

```
1: Compute the WSPD on the given point set.
2: For two points of each well-separated pair, approximate their minimum-perimeter triangle
   (see Lemma 2.1).
3: Construct an undirected graph whose edge set consists of edges of approximated triangles.
4: Orient the undirected graph using a greedy algorithm (see Lemma 2.2).
```

In Step 2, we approximate the minimum-perimeter triangle $\Delta^*(a,b)$ of two points $a \in A$ and $b \in B$ of a well-separated pair $\{A, B\}$. We claim that any third point c in either A or B forms a triangle Δ_{abc} that is a $(1+\varepsilon)$-approximation of $\Delta^*(a,b)$.

If both subsets have size one, thus no such c exists, we approximate the minimum triangle by approximate nearest neighbour queries. Therefore, we use the data structure of Arya, Mount, Netanyahu, Silverman and Wu [2], which can be constructed in $\mathcal{O}(n \log n)$ time and has size $\mathcal{O}(n)$. Given a real number $\varepsilon > 0$ and a query point q in $\mathbb{R}^d$, we compute a $(1+\varepsilon)$-approximate nearest neighbour of q in P in $\mathcal{O}(\log n)$ time. Given two points $p, q \in P$ and the real numbers $\varepsilon_2 = \varepsilon_1/2$, $\alpha = 4/\varepsilon_1$, the following query returns a $(1+\varepsilon_1)$-approximation of $\Delta^*(p,q)$:

Algorithm 2 $(1+\varepsilon_1)$-Approximation of the Minimum-Perimeter Triangle

```
1: Compute a $(1+\varepsilon_2)$-approximate nearest neighbour $r$ of $p$ (which is neither $p$ nor $q$).
2: If $|pr| > \alpha|pq|$, then return $\Delta_{pqr}$.
3: Otherwise, let $B$ be the hypercube with sides of length $3\alpha|pq|$ that is centred at $p$ and
   divided into cells with sides of length of $\frac{2}{3\sqrt{d}} \cdot \varepsilon_1|pq|$.
4: For each cell, compute the $(1+\varepsilon_2)$-approximate nearest neighbour $x_c$ of the cell centre $c$
   (which is neither $p$ nor $q$).
5: Return the triangle $\Delta_{pqx_c}$ with the smallest perimeter across all cells.
```

▶ **Lemma 2.1.**[*] *Let P be a set of n points in $\mathbb{R}^d$ and let $0 < \varepsilon_1 < 2$ be a real number. In $\mathcal{O}(n \log n)$ time, the set P can be preprocessed into a data structure of size $\mathcal{O}(n)$, such that, given any two distinct query points p and q in P, a $(1+\varepsilon_1)$-approximation of the minimum triangle $\Delta^*(p,q)$ can be computed in $\mathcal{O}(\log n)$ time.*

In Step 4 of Algorithm 1, we modify a greedy algorithm from [4], which originally works in the following way. Sort the $\binom{n}{3}$ triangles of the complete graph in ascending order by their perimeter. For each triangle in this order, orient it clockwise or anti-clockwise if possible; otherwise, we skip this triangle. At the end of the algorithm, all remaining edges are oriented

[*] Due to space restrictions, proofs of results marked by $*$ are given in the full version of this paper.

arbitrarily. In [4], they show that this greedy algorithm yields an oriented 2-spanner. Note that this graph has $\binom{n}{2}$ edges. Our modification of the greedy algorithm is to only consider the approximate minimum-perimeter triangles, rather than all $\binom{n}{3}$ triangles. Lemma 2.2 states that, for any two points whose triangles are $(1+\varepsilon_1)$-approximated in Step 2 of Algorithm 1, the dilation between those points is at most $(2+2\varepsilon_1)$ in the resulting orientation.

▶ **Lemma 2.2.*** *Let P be a set of n points in $\mathbb{R}^d$, let $\varepsilon_1 > 0$ be a real number, and let L be a list of m point triples $(p,q;r)$, with p, q, and r being pairwise distinct points in P. Assume that for each $(p,q;r)$ in L, $|\Delta_{pqr}| \le (1+\varepsilon_1)\cdot|\Delta^*(p,q)|$.*

Let $G=(P,E)$ be the undirected graph whose edge set consists of all edges of the m triangles Δ_{pqr} defined by the triples $(p,q;r)$ in L. In $\mathcal{O}(m\log n)$ time, we can compute an orientation $\overrightarrow{G}$ of G, such that $\mathrm{odil}_{\overrightarrow{G}}(p,q) \le 2+2\varepsilon_1$ for each $(p,q;r)$ in L.

▶ Remark. The above lemma only guarantees an upper bound on $\mathrm{odil}_{\overrightarrow{G}}(p,q)$ for $(p,q;r)$ in L. For such a triple, $\mathrm{odil}_{\overrightarrow{G}}(p,r)$ and $\mathrm{odil}_{\overrightarrow{G}}(q,r)$ can be arbitrarily large.

Now, we show that the algorithm described above constructs a sparse oriented $(2+\varepsilon)$-spanner for point sets in the Euclidean space $\mathbb{R}^d$:

▶ **Theorem 2.3.*** *Given a set P of n points in $\mathbb{R}^d$ and a real number $0<\varepsilon<2$, an oriented $(2+\varepsilon)$-spanner for P with $\mathcal{O}(n)$ edges can be constructed in $\mathcal{O}(n\log n)$ time and $\mathcal{O}(n)$ space.*

Proof sketch. We show that Algorithm 1 returns such a spanner $\overrightarrow{G}$. For any two points $p,q\in P$, we prove that $|C_{\overrightarrow{G}}(p,q)| \le (2+\varepsilon)\cdot|\Delta^*(p,q)|$ by induction on the rank of $|pq|$ in the sorted sequence of all $\binom{n}{2}$ pairwise distances.

If p,q is a closest pair (for the definition see [17]), then $\{\{p\},\{q\}\}$ is in the WSPD. Therefore, a $(1+\varepsilon_1)$-approximation of $\Delta^*(p,q)$ is added to the list of triangles, that define the edge set of $\overrightarrow{G}$ and, due to Lemma 2.2 and for $\varepsilon_1=\varepsilon/4$, we obtain $\mathrm{odil}_{\overrightarrow{G}}(p,q)\le 2+2\varepsilon_1 \le 2+\varepsilon$.

Assume that p,q is not a closest pair. Let $\{A,B\}$ be the well-separated pair with $p\in A$ and $q\in B$. In Step 2 of Algorithm 1, one point $a\in A$ and one point $b\in B$ are used to approximate a minimum-perimeter triangle. We now sketch the case, where $p\ne a$ and $q\ne b$ (see Figure 3). In particular, we show

$$|C_{\overrightarrow{G}}(p,q)| \le |C_{\overrightarrow{G}}(p,a)| + |C_{\overrightarrow{G}}(a,b)| + |C_{\overrightarrow{G}}(b,q)| \le (2+\varepsilon)\cdot|\Delta^*(p,q)|.$$

For the other cases, where $p=a$ or $p=b$, this statement also applies.

Since $|pa|<|pq|$ and $|bq|<|pq|$, we use induction to bound $|C_{\overrightarrow{G}}(p,a)|$ and $|C_{\overrightarrow{G}}(b,q)|$. Further, Lemma 2.2 applies to a and b, thus:

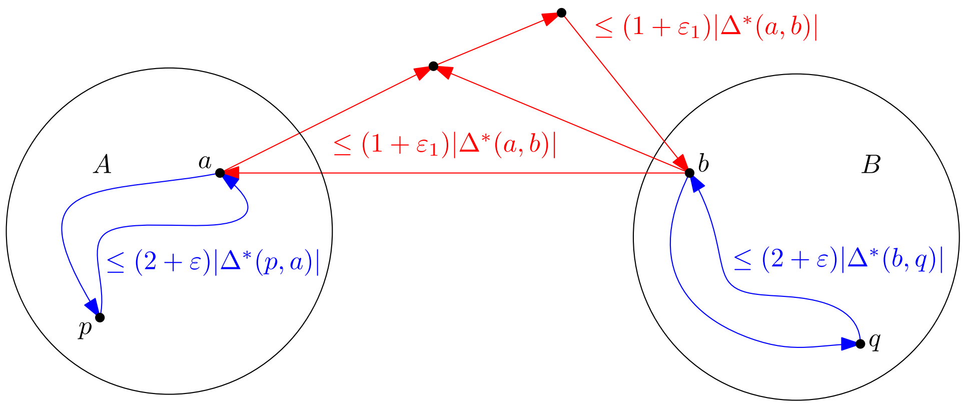
$$|C_{\overrightarrow{G}}(p,q)| \le (2+\varepsilon)|\Delta^*(p,a)| + (2+2\varepsilon_1)|\Delta^*(a,b)| + (2+\varepsilon)|\Delta^*(b,q)|.$$

Let $s>0$ be the real number, such that $\{A,B\}$ is an s-well-separated pair. Using properties of the WSPD, if $|A|\ge 3$ and $|B|\ge 3$, we can bound the perimeter of $\Delta^*(p,a)$ and $\Delta^*(b,q)$ by $3/s|\Delta^*(p,q)|$. Further, $|\Delta^*(a,b)|$ is bounded by $6/s|\Delta^*(p,q)$.

For $\varepsilon_1=\varepsilon/4$, $\varepsilon<2$ and $s=96/\varepsilon$, we conclude that $|C_{\overrightarrow{G}}(p,q)|\le(2+\varepsilon)|\Delta^*(p,q)|$. ◀

3 Oriented Dilation

Computing the dilation of a given graph is related to the all-pairs shortest paths problem. There is a well-known cubic-time algorithm for APSP by Floyd and Warshall [9, 18]. Using their algorithm (and computing the minimum triangles naively), the oriented dilation of a given graph with n points can be computed in $\mathcal{O}(n^3)$ time.



$\leq (1+\varepsilon_1)|\Delta^*(a,b)|$
$\leq (1+\varepsilon_1)|\Delta^*(a,b)|$
A
a
b
B
$\leq (2+\varepsilon)|\Delta^*(p,a)|$
$\leq (2+\varepsilon)|\Delta^*(b,q)|$
p
q

Figure 3 Visualization of Case 1: $|C_{\overrightarrow{G}}(p,q)| \leq |C_{\overrightarrow{G}}(p,a)| + |C_{\overrightarrow{G}}(a,b)| + |C_{\overrightarrow{G}}(b,q)|$ in the proof of Theorem 2.3. $C_{\overrightarrow{G}}(a,b)$ is either a $(1+\varepsilon_1)$-approximation of $\Delta^*(a,b)$ or a union of two triangles (red), each of at most $(1+\varepsilon_1)|\Delta^*(a,b)|$. $|C_{\overrightarrow{G}}(p,a)|$ and $|C_{\overrightarrow{G}}(b,q)|$ (blue) are bounded inductively.

We prove APSP-hardness for computing the oriented dilation for a given graph. Since the APSP-problem is believed to need truly cubic time [15], it is unlikely that computing the oriented dilation is possible in subcubic time. Note that the following statement applies to *metric graphs*, thus we have general metric distances instead of Euclidean.

▶ **Theorem 3.1.*** *Let $\overrightarrow{G}$ be an oriented metric graph. It is APSP-hard to compute the oriented dilation of $\overrightarrow{G}$.*

To compute the oriented dilation, testing all $\binom{n}{2}$ point tuples seems to be unavoidable. However, to approximate the oriented dilation, we show that, as for approximating undirected dilation (see [12, 14] for a survey), a linear set of tuples suffices. To complete this, we give the following approximation algorithm (a detailed version is given in the full version):

Algorithm 3 Oriented Dilation Approximation

```
1: Compute the WSPD on the point set of the given graph $\overrightarrow{G}$.
2: For two points of each well-separated pair, approximate their shortest closed walk in $\overrightarrow{G}$
   and their minimum-perimeter triangle. Compute the division of both.
3: Return the smallest approximated oriented dilation across all tuples.
```

▶ **Theorem 3.2.*** *Let P be a set of n points in $\mathbb{R}^d$, let $\overrightarrow{G}$ be an oriented graph on P, and let $\varepsilon > 0$ be a real number. Assume that, in $T(n)$ time, we can construct a data structure such that, for any two query points p and q in P, we can return, in $f(n)$ time, a k-approximation to the length of a shortest path in G between p and q. Then we can compute a value $\overline{\text{odil}}$ in $\mathcal{O}(T(n) + n(\log n + f(n)))$ time, such that*

$$(1-\varepsilon) \cdot \text{odil}(\overrightarrow{G}) \leq \overline{\text{odil}} \leq k \cdot \text{odil}(\overrightarrow{G}).$$

The precision and runtime of this approximation depend on approximating the length of a shortest path between two query points. The point-to-point shortest path problem in directed graphs is an extensively studied problem (see [11] for a survey). Alternatively, shortest path queries can be preprocessed via approximate APSP (see [8, 16, 19]). For planar directed graphs, the authors in [1] present a data structure build in $\mathcal{O}(n^2/r)$ time, such that the exact length of a shortest path between two query points can be returned in $\mathcal{O}(\sqrt{r})$ time. Setting $r = n^{2/3}$ minimizes the runtime of our Algorithm 3 to $\mathcal{O}(n^{4/3})$. It should be noted that even with an exact shortest path queries ($k = 1$) the oriented dilation remains an approximation due to the minimum-perimeter triangle approximation.

4 Conclusion and Outlook

We present an algorithm for constructing a sparse oriented $(2 + \varepsilon)$-spanners for multidimensional point sets. Our algorithm computes such a spanner efficiently in $O(n \log n)$ time for point sets of size n. In contrast, [4] presents a plane oriented $\mathcal{O}(1)$-spanner, but only for points in convex position. Developing algorithms for constructing plane oriented $\mathcal{O}(1)$-spanners for general two-dimensional point sets remains an open problem. Another natural open problem is to improve upon $t = 2 + \varepsilon$. For this, the question arises: Does every point set admit an oriented t-spanner with $t < 2$? Even for complete oriented graphs this is open.

In the second part of this paper, we study the problem of computing the oriented dilation of a given graph. We prove APSP-hardness of this problem for metric graphs. We complement this by a subcubic approximation algorithm. The APSP-hardness of computing the oriented dilation of metric graphs seems to be dominated by the computation of a minimum-perimeter triangle. However, for Euclidean graphs, the minimum-perimeter triangle containing two given points can be computed in $o(n)$ time. This raises the question: Is computing the oriented dilation of a Euclidean graph easier than for general metric graphs?

References

1 Srinivasa Rao Arikati, Danny Z. Chen, L. Paul Chew, Gautam Das, Michiel H. M. Smid, and Christos D. Zaroliagis. Planar spanners and approximate shortest path queries among obstacles in the plane. In Josep Díaz and Maria J. Serna, editors, *Proc. 4th Annu. European Sympos. Algorithms (ESA)*, volume 1136 of *Lecture Notes in Computer Science*, pages 514–528. Springer-Verlag, 1996. doi:10.1007/3-540-61680-2_79.

2 Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998. doi:10.1145/293347.293348.

3 Prosenjit Bose and Michiel H. M. Smid. On plane geometric spanners: A survey and open problems. *Comput. Geom. Theory Appl.*, 46(7):818–830, 2013. doi:10.1016/j.comgeo.2013.04.002.

4 Kevin Buchin, Joachim Gudmundsson, Antonia Kalb, Aleksandr Popov, Carolin Rehs, André van Renssen, and Sampson Wong. Oriented spanners. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *Proc. 31st Annu. European Sympos. Algorithms (ESA)*, volume 274 of *LIPIcs*, pages 26:1–26:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ESA.2023.26.

5 Kevin Buchin, Antonia Kalb, Guangping Li, and Carolin Rehs. Experimental analysis of oriented spanners on one-dimensional point sets. In *Proc. 36th Canad. Conf. Comput. Geom. (CCCG)*, pages 223–231, 2024.

6 Kevin Buchin, Antonia Kalb, Carolin Rehs, and André Schulz. Oriented dilation of undirected graphs. In *30th European Workshop on Computational Geometry (EuroCG)*, pages 65:1–65:8, 2024.

7 Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. ACM*, 42(1):67–90, 1995. doi:10.1145/200836.200853.

8 Michal Dory, Sebastian Forster, Yael Kirkpatrick, Yasamin Nazari, Virginia Vassilevska Williams, and Tijn de Vos. Fast 2-approximate all-pairs shortest paths. In *Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 4728–4757. SIAM, 2024. `doi:10.1137/1.9781611977912.169`.

9 Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5:345 – 345, 1962. `doi:10.1145/367766.368168`.

10 Panos Giannopoulos, Rolf Klein, Christian Knauer, Martin Kutz, and Dániel Marx. Computing geometric minimum-dilation graphs is NP-hard. *Internat. J. Comput. Geom. Appl.*, 20(2):147–173, 2010. `doi:10.1142/S0218195910003244`.

11 Andrew V. Goldberg and Chris Harrelson. Computing the shortest path: A search meets graph theory. In *Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 156–165, 2005. `doi:10.5555/1070432.1070455`.

12 Joachim Gudmundsson and Christian Knauer. Dilation and detours in geometric networks. In *Handbook of Approximation Algorithms and Metaheuristics*, pages 53–69. Chapman and Hall/CRC, 2018. `doi:10.1201/9781420010749-62`.

13 J. Mark Keil and Carl A. Gutwin. Classes of graphs which approximate the complete euclidean graph. *Discret. Comput. Geom.*, 7:13–28, 1992. `doi:10.1007/BF02187821`.

14 Giri Narasimhan and Michiel H. M. Smid. *Geometric Spanner Networks.* Cambridge University Press, 2007. `doi:10.1017/CBO9780511546884`.

15 K. R. Udaya Kumar Reddy. A survey of the all-pairs shortest paths problem and its variants in graphs. *Acta Univ. Sapientiae Inform*, 8:16–40, 2016. `doi:10.1515/ausi-2016-0002`.

16 Barna Saha and Christopher Ye. Faster approximate all pairs shortest paths. In *Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 4758–4827. SIAM, 2024. `doi:10.1137/1.9781611977912.169`.

17 Michiel H. M. Smid. The well-separated pair decomposition and its applications. In *Handbook of Approximation Algorithms and Metaheuristics (2)*. Chapman and Hall/CRC, 2018.

18 Stephen Warshall. A theorem on boolean matrices. *J. ACM*, 9(1):11–12, 1962. `doi:10.1145/321105.321107`.

19 Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *J. ACM*, 49(3):289–317, 2002. `doi:10.1145/567112.567114`.

A Linear Time Algorithm for Finding Minimum Flip Sequences between Plane Spanning Paths in Convex Point Sets*

Oswin Aichholzer[1] **and Joseph Dorfer**[2]

1 **Graz University of Technology**
`oswin.aichholzer@tugraz.at`

2 **Graz University of Technology**
`joseph.dorfer@tugraz.at`

Abstract

We provide a linear time algorithm to determine a minimum flip sequence between two plane spanning paths on a point set in convex position. In contrast, we show that the happy edge property does not hold in this setting. This differs from several results on reconfiguration problems where the absence of the happy edge property implies algorithmic hardness of the problem.

1 Introduction

Let S be a finite point set in the plane in general position, that is, no three points lie on a common line. We call S a *convex* point set if no point in S lies in the interior of the convex hull of S. For a convex point set with n vertices label the points v_0,...,v_{n-1} in clockwise order starting from an arbitrary vertex. A plane straight-line graph on S is a graph with vertex set S and whose edges are straight line segments between pairs of points of S such that no two edges intersect except at a common endpoint. Throughout this paper all graphs are assumed to be straight line graphs.

Flips in Plane Spanning Paths. A *flip* is an operation that removes one edge from a plane spanning path and adds another edge such that the resulting structure is again a plane path. Given an initial plane spanning path P_{in} and a target path P_{tar} on S, a *flip sequence* from P_{in} to P_{tar} is a sequence of plane paths P_0, P_1, ..., P_k such that $P_0 = P_{in}$, $P_k = P_{tar}$ and two consecutive paths differ only by a flip. The index k describes the *length* of the flip sequence. The *flip distance* between P_{in} and P_{tar} is the minimum k for which a flip sequence from P_{in} to P_{tar} of length k exists. A flip sequence that realizes this minimum will be called a *minimum flip sequence.*

In [1] a characterization of flips in plane spanning paths into three types is given. See Figure 1 for an illustration. Consider a path $P = p_1p_2...p_n$ where the p_i denotes the points in order of traversal by the path. A flip of *Type 1* removes an edge (p_{i-1}, p_i) and adds an edge (p_1, p_i) or (p_{i-1}, p_n) resulting in a new path, $p_{i-1}, ..., p_1p_i, ..., p_n$ or $p_1, ..., p_{i-1}p_n, ..., p_i$, respectively. A flip of *Type 2* adds the edge (p_1, p_n) assuming this edge does not cross any of the already existing edges. Afterwards, an arbitrary edge (p_{i-1}, p_i) from the original path can be removed. Note that adding and removing consecutive edges when closing a cycle can be interpreted as both a Type 1 flip and a Type 2 flip. For simplicity of notation we will count such flips as Type 2 flips. A flip of *Type 3* also adds the edge (p_1, p_n) but now it is

* This research was funded in part by the Austrian Science Fund (FWF) 10.55776/DOC183.

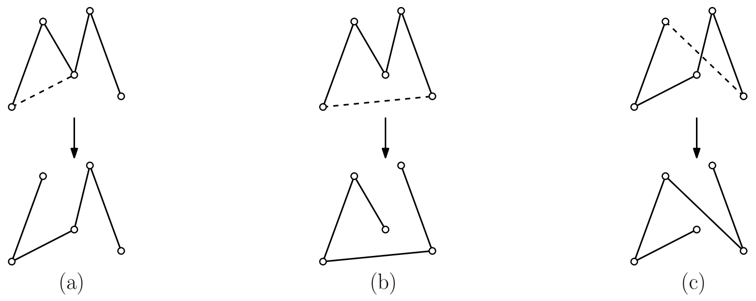


Figure 1 Flips in plane spanning paths: (a) Type 1 flip (b) Type 2 flip (c) Type 3 flip

assumed to intersect exactly one edge (p_{i-1}, p_i) in P which is then removed within this flip operation.

It is still unknown, whether every plane spanning path can be transformed into any other spanning path on the same point set. But, for the special cases of convex point sets [3], wheel sets and generalized double circles [1], and point sets with at most two convex layers [6], it is shown that we can always flip from one path to another. In [6] the existance of a large connected component of plane spanning paths called suffix-independent paths is derived.

Happy Edges. *Happy edges* are edges that lie in both, the initial configuration and the target configuration of a graph reconfiguration problem. The so-called *happy edge property* says that there always exists a minimum flip sequence between two configurations in which no edge is removed and later flipped in again. This implies that a happy edge is never flipped in such a sequence. The happy edge property often is a good indication for the complexity of a reconfiguration problem. For example, for triangulations of simple polygons [2] and general point sets [7, 8] finding minimum flip sequences is NP-hard, while the gadgets in the proofs are built around conterexamples to the happy edge property. Conversely, the happy edge property holds for plane perfect matchings of convex point sets and a minimum flip sequence can be found in polynomial time [5]. On the other hand, the happy edge property is known to hold for triangulations of convex polygons [9], but the complexity of finding minimum flip sequences is still open.

Our Contributions. In this paper we show that the happy edge property does not hold for plane spanning paths of convex point sets. This adds to the already known counterexamples in general point sets [4]. In contrast, we provide an approach for finding minimum flip sequences between pairs of plane spanning paths on convex point sets in linear time. Interestingly, this differs from all the previously mentioned results where the absence of the happy edge property implied hardness of finding minimum flip sequences.

2 Counterexample to the Happy Edge Property

In Figure 2 we show the flip graph of all eight plane spanning paths on four points in convex position. The initial path P_{in} and target path P_{tar} are marked. Both paths share a diagonal, but no other path contains that particular diagonal. Moreover, the two paths can not be directly transformed into one another via one flip. Therefore, this pair of paths provides a counterexample to the happy edge property for finding minimum flip sequences between plane spanning paths in convex point sets.

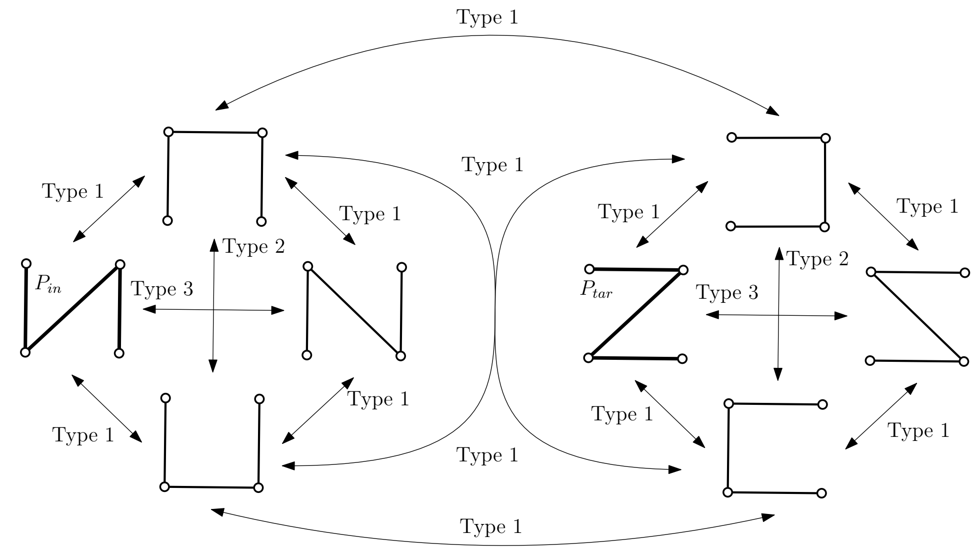


Figure 2 Flip graph of paths on four points in convex position with a counterexample to the happy edge property. Indicated are the different types of flips and the initial and target paths.

3 Basics on Flips in Paths

In this section we show some basic relations for flips between plane paths. Due to visibility constraints Type 2 flips can only happen when all edges of the current path lie on the convex hull. For Type 1 flips we obtain the following result.

▶ **Lemma 3.1.** *Type 1 flips either remove a diagonal or add a diagonal, but not both at the same time. If diagonals exists, we can always lower the number of diagonals via a Type 1 flip.*

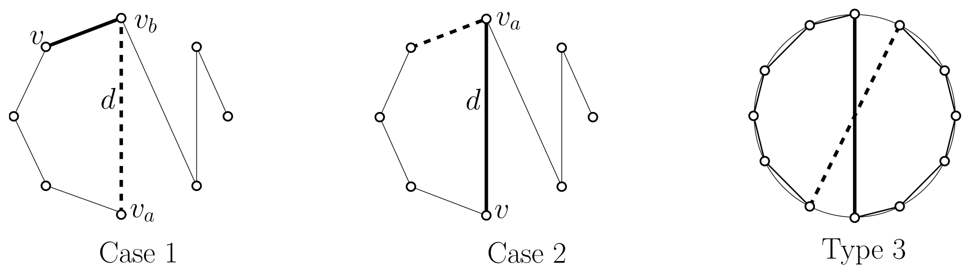


Figure 3 Case 1 (left): Adding a convex hull edge (fat), removing a diagonal (dashed). Case 2 (middle): Adding a diagonal (fat), removing a convex hull edge (dashed). For a Type 3 flip (right), we need all but one diagonal to be on the convex hull.

A proof for Lemma 3.1 can be found in a full version of this paper. The intuition behind Lemma 3.1 can be seen in Figure 3. It shows all possible flips that can involve diagonals: Removing the unique diagonal that is visible from an end vertex of the path by adding a convex hull edge between this end vertex and a vertex incident to the diagonal (Case 1), adding a diagonal by removing a convex hull edge (Case 2), and performing a Type 3 flip that exchanges the only diagonal with a diagonal that is rotated by one vertex into some direction (right).

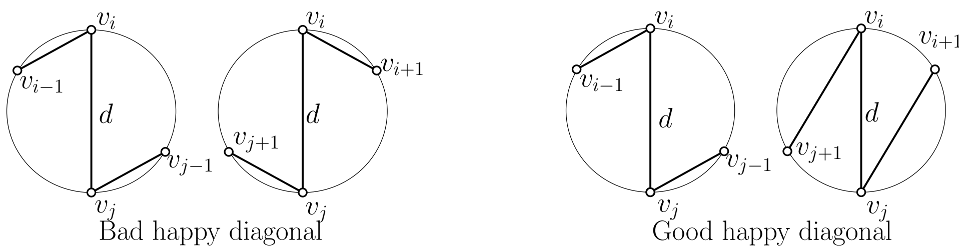


Figure 4 Bad happy diagonal D with edges emanating to different sides in different paths (left), and good happy diagonal d with edges emanating to the same side in different paths (right)

Let $d = (v_i, v_j)$ be a happy edge that is a diagonal. Note that d splits the convex point set into two parts and its two adjacent edges emanate into different parts. Let d be adjacent to e_1 at v_i and f_1 at v_j in the initial path P_{in}, and to e_2 at v_i and f_2 at v_j in the target path P_{tar}. We call d a *good happy diagonal* if e_1 and e_2 emanate to the same side of d, that is, the second vertices of e_1 and e_2 are either both in $\{v_{i+1}, v_{j-1}\}$ or both in $\{v_{j+1}, v_{i-1}\}$. Otherwise, we call d a *bad happy diagonal* if e_1 and e_2 emanate to different sides (see Figure 4).

▶ **Lemma 3.2.** *Consider two paths P_{in} and P_{tar} as defined above.*

(a) For every flip sequence from P_{in} to P_{tar} any bad happy diagonal d needs to be removed.

(b) For every subpath S of consecutive happy edges that contains at least one good happy diagonal there exists a flip sequence from P_{in} to P_{tar} that preserves all edges of S.

▶ **Proof.** For (a) observe that we cannot exchange edges e_1 and f_1 with e_2 and f_2 in one flip.

For (b) start with P_{in}, pick one end vertex of P_{in} and perform flips that remove diagonals until the next diagonal that is visible from the current endvertex lies in S. Then, repeat the process for the other endvertex of P_{in}. The resulting path P' will then consist of S and two (possibly empty) subpaths of convex hull edges. Do the same for P_{tar} to obtain a path P''. Since S contains a good happy diagonal and there are only two ways how the subpath of convex hull edges may look like, we get $P' = P''$. The required flip sequence from P_{in} to P_{tar} consists of the flips from P_{in} to P' and the flips from P_{tar} to P'' in reverse order. For an example of such a flip sequence, see Figure 5. ◀

Note that if a subpath of happy edges of length at least 2 contains a diagonal, the diagonal is already a good happy diagonal.

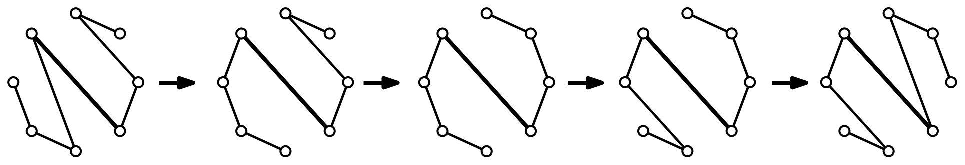


Figure 5 Illustration of Lemma 3.2(b): A flip sequence that preserves a good happy diagonal

4 Characterization of Minimum Flip Sequences

Based on the structure of good happy diagonals and convex hull edges of the initial path and the target path, we provide a characterization of pairs of paths into four categories. We derive lower bounds on the number of flips and argue that for each category there exists a flip sequence that makes exactly this number of flips, thus providing a minimal flip sequence.

▶ **Theorem 4.1.** *Let P_{in} and P_{tar} be two plane spanning paths for the same convex point set. Let k and l denote the number of diagonals of P_{in} and P_{tar}, respectively. Then the flip distance between P_{in} and P_{tar} is described by the following cases.*

- ***Case 1:*** *If good happy diagonals exist, let $m \geq 1$ be the maximal number of good happy diagonals in a subpath of consecutive happy edges. Then the flip distance is $k + l - 2m$.*
- ***Case 2:*** *If no good happy diagonal exists we distinguish three cases*

  Case 2a: *If there exists a pair of diagonals $d_1 \in P_{in}$, $d_2 \in P_{tar}$ that can eventually be exchanged for one another by a Type 3 flip, then the flip distance is $k + l - 1$.*

  Case 2b: *If no Type 3 flip can be performed and if the diagonals can be flipped to convex hull edges in both, the initial and target path, such that the paths after flipping all diagonals coincide, the flip distance is $k + l$.*

  Case 2c: *Otherwise the flip distance is $k + l + 1$.*

▶ **Remark 4.2.** *Regarding Case 2b: If a path P does not contain the convex hull edge (v_i, v_{i+1}), we say that P has a* gap *g in the convex hull at (v_i, v_{i+1}). If g is a gap of both paths, P_{in} and P_{tar}, we say that P_{in} and P_{tar} have a* common gap *in the convex hull at g. Observe that by removing diagonals we can flip a path P into a path that contains all convex hull edges except for the gap g if and only if g is a gap of P. This can be done by adding convex hull edges incident to each end vertex and removing diagonals until the next added convex hull edge would lie in g. For an intuitive example see Figure 6.*

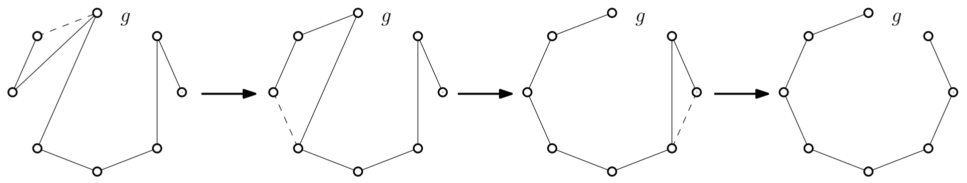


Figure 6 Illustration of Remark 4.2. In every step the to be added convex hull edge is indicated via a dashed line. First diagonals to the left of the gap g are removed by the left end vertex of the path. Afterwards, the diagonal to the right gets removed by the right end vertex of the path.

▶ **Proof** (Theorem 4.1)**.** For a visualization of the different cases, see Figure 7.

Case 1: Consider a decomposition of the point set where points belong to the same component if and only if they lie on the same side of every good happy diagonal. Then, there are exactly two components that each contain one of the endpoints of the path. Therefore, to remove and add edges in components that do not contain endpoints, we need to remove all good happy diagonals to one side of this component. From this, we get that we need to remove the good happy diagonals from all but one subpath of happy edges. Let m_S denote the number of good happy diagonals in a subpath of happy edges S.

By Lemma 3.1 and Lemma 3.2, removing all diagonals apart from the ones in S takes at least $k - m_S$ flips and can be done in that number of flips. Similarly, adding all the new diagonals takes $l - m_S$ flips. Therefore, there is a flip sequence from P_{in} to P_{tar} that preserves S with $k + l - 2m_S$ flips. The length of the minimum flip sequence is therefore $k + l - 2m$ by the choice of m and it is attained by applying the flips from the proof of Lemma 3.2.

Case 2a: The only way to exchange more than one diagonal at once is by performing a Type 3 flip. If we want to exchange one diagonal from P_{in} directly with a diagonal in P_{tar}, all the flips leading up to the Type 3 flip need to remove the $k - 1$ diagonals that are not involved in the Type 3 flip from the initial path. Similarly, all the flips that occur after the Type 3 flip add the $l - 1$ diagonals of the target path that are not involved in the Type 3 flip. Further, the two subpaths of convex hull edges need to coincide between the paths before and after the Type 3 flip, see Figure 3 (right). Therefore, the subpaths of convex hull edges are already correctly aligned after the Type 3 flip. This shows that $k + l - 1$ flips are necessary and sufficient in case a Type 3 flip can occur, $k - 1$ for removing diagonals from P_{in}, one Type 3 flip and $l - 1$ flips to add diagonals to get P_{tar}.

Case 2b: If no Type 3 flip can be set up, it follows from Lemma 3.1 that k flips are necessary and sufficient to remove all diagonals from the initial path, and similarly l flips to add all the diagonals of the new path. Since the diagonals can be removed in a way that there is a common gap in the convex hull, $k + l$ flips are indeed necessary and sufficient.

Case 2c: If neither Case 2a and Case 2b hold, so in particular P_{in} and P_{tar} do not have a common gap in the convex hull, it is indeed necessary to remove all diagonals, realign the convex hull and add all diagonals to get the new path. So $k + l + 1$ flips are necessary.

To show optimality also for Case 2c assume we could start adding diagonals from the target path before removing all diagonals from the initial path and realigning the path along the convex hull. We add the first diagonal $d = (v_i, v_j) \in P_{tar}$ in the step from P_l to P_{l+1}. Assume v_i was the end vertex in the previous step. By the assumption of Case 2c, v_i is incident to two convex hull edges in $P_{in} \cup P_{tar}$. One of them, say e_1, is in $P_{tar} \setminus P_{in}$, otherwise there would not be an isolated vertex at v_i. Since v_i has degree at most two in P_{tar} the other convex hull edge, say e_2, has to be in $P_{in} \setminus P_{tar}$. Since we didn't remove any convex hull edge from P_{in} prior to that flip, the edges incident to v_i, e_1 and e_2 emanate to different sides of d in P_{l+1} and P_{tar}. Therefore, d is a bad happy diagonal and has to be removed again by Lemma 3.2a. ◀

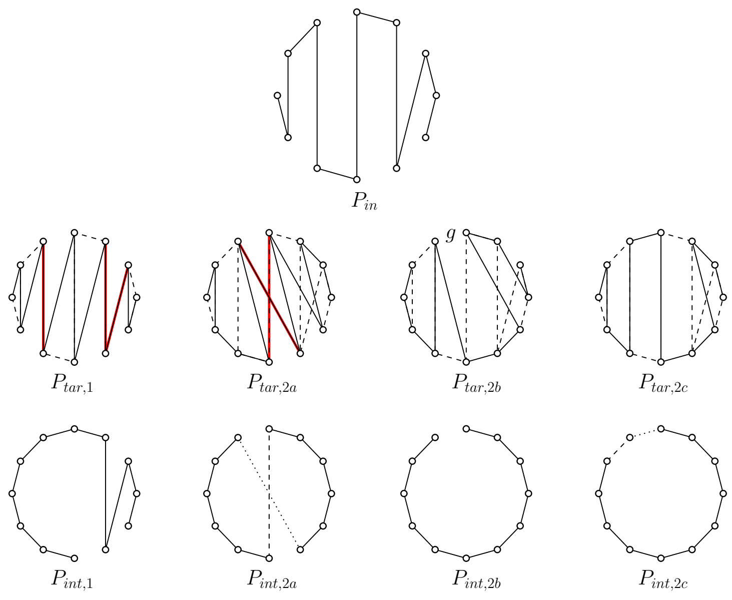


Figure 7 Visualization of Theorem 4.1: Top row: the initial path P_{in}. Second row: target paths where dashed lines hint at the initial path. The cases 1, 2a, 2b and 2c occur from left to right. In Case 1 good happy edges are marked in red. In Case 2b the two edges involved in a Type 3 flip are marked red. For Case 2b we labeled the common gap g in the convex hull. Observe that for Case 2c, none of the other cases occur. Bottom row: intermediate configurations for the corresponding cases.

The exact details on how to implement all the checks of Theorem 4.1 to run in linear time can be found in a full version of this paper.

▶ **Theorem 4.3.** *The flip distance between two plane spanning paths in convex point sets can be determined in $O(n)$ time and space.*

5 Conclusion

We disproved the happy edge property for plane spanning paths on convex point sets. At the same time, we provided a linear time algorithm to compute the shortest flip sequence between two given plane spanning paths. This contradicts the assumption that the absence of the happy edge property makes finding minimum flip sequences hard to solve. We are not aware of any problem for which the opposite direction fails, that is, a reconfiguration problem for which the happy edge property is true, but it is hard to find minimum flip sequences.

Two interesting open problems are:

1. Can we flip between any two plane spanning paths for point sets in general position?
2. If the answer to the previous question is yes, how many flips does it take in the worst case and what is the complexity of computing the flip distance between two given paths?

Acknowledgements

The results of this work were obtained during the PAGCAP workshop in Strobl in May 2024. We want to thank the participants of the workshop for fruitful discussions and a productive environment. Further, we want to thank the anonymous referees for helpful remarks.

References

1 Oswin Aichholzer, Kristin Knorr, Wolfgang Mulzer, Johannes Obenaus, Rosna Paul, and Birgit Vogtenhuber. Flipping plane spanning paths. In Chun-Cheng Lin, Bertrand M. T. Lin, and Giuseppe Liotta, editors, *WALCOM: Algorithms and Computation*, pages 49–60, Cham, 2023. Springer Nature Switzerland. arXiv:2202.10831, doi:10.1007/978-3-031-27051-2_5.

2 Oswin Aichholzer, Wolfgang Mulzer, and Alexander Pilz. Flip distance between triangulations of a simple polygon is NP-complete. *Discrete & Computational Geometry*, 54(2):368–389, June 2015. doi:10.1007/s00454-015-9709-7.

3 Jou-Ming Chang and Ro-Yu Wu. On the diameter of geometric path graphs of points in convex position. *Information Processing Letters*, 109(8):409–413, 2009. doi:10.1016/j.ipl.2008.12.017.

4 Stefan Felsner. Personal communication.

5 M. Carmen Hernando, Fabrizio Hurtado, and Marc Noy. Graphs of non-crossing perfect matchings. *Graphs and Combinatorics*, 18:517–532, 01 2002. doi:10.1007/s003730200038.

6 Linda Kleist, Peter Kramer, and Christian Rieck. On the connectivity of the flip graph of plane spanning paths, 2024. arXiv:2407.03912.

7 Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Computational Geometry*, 49:17–23, 2015. arXiv:1205.2425, doi:10.1016/j.comgeo.2014.11.001.

8 Alexander Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 47(5):589–604, 2014. doi:10.1016/j.comgeo.2014.01.001.

9 Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *Journal of the American Mathematical Society*, 1(3):647–681, July 1988. doi:10.1090/S0894-0347-1988-0928904-4.

Geometric spanners of bounded tree-width

Kevin Buchin[1], **Carolin Rehs**[1], **and Torben Scheele**[1]

1 **TU Dortmund, Germany**
`kevin.buchin@tu-dortmund.de`, `carolin.rehs@tu-dortmund.de`,
`torben.scheele@tu-dortmund.de`

Abstract

Given a point set P in the Euclidean space, a geometric t-spanner G is a graph on P such that for every pair of points, the shortest path in G between those points is at most a factor t longer than the Euclidean distance between those points. The value $t \geq 1$ is called the dilation of G. Commonly, the aim is to construct a t-spanner with additional desirable properties. In graph theory, a powerful tool to admit efficient algorithms is bounded tree-width. We investigate the problem of computing geometric spanners with bounded tree-width and small dilation t.

Let d be a fixed integer and $P \subset \mathbb{R}^d$ be a point set with n points. We give a first algorithm to compute an $\mathcal{O}(n/k^{d/(d-1)})$-spanner on P with tree-width at most k. The dilation obtained by the algorithm is asymptotically worst-case optimal for graphs with tree-width k: We show that there is a set of n points such that every spanner of tree-width k has dilation $\mathcal{O}(n/k^{d/(d-1)})$. We further prove a tight dependency between tree-width and the number of edges in sparse connected planar graphs, which admits, for point sets in $\mathbb{R}^2$, a plane spanner with tree-width at most k and small maximum vertex degree.

Related Version *Full Version*: `https://arxiv.org/abs/2412.06316`

1 Introduction

Geometric spanners are an extensively studied area in computational geometry, see [4, 13] for surveys. A geometric spanner for a point set $P \in \mathbb{R}^d$ is a weighted graph on P, where the weight of an edge is the Euclidean distance between its endpoints, aiming for sparsity while avoiding long paths between two points in P. This is formalised by the dilation $t = \max\left\{ \frac{d(p,p')}{|pp'|} \;\middle|\; p, p' \in P,\ p \neq p' \right\}$, where $d(p,p')$ is the distance between two points in G and $|pp'|$ is the Euclidean distance between p and p'. A geometric spanner with dilation at most t is also called a *t-spanner*.

When geometric spanners were introduced in 1986 [5], sparsity was obtained by requiring planarity. Since then, plane geometric spanners have been widely studied [4]. But also other measures are of interest: Especially spanners with a linearly bounded number of edges or a bounded vertex degree have been widely researched [10, 13, 18, 8, 3, 6, 9, 2].

In graph theory, an important tool to design efficient algorithms for problems that are NP-hard in general is to restrict these problems to graphs which are bounded in certain parameters. Especially graphs with bounded *tree-width* [14, 15] allow polynomial-time algorithms for many in general NP-hard problems. Thus, geometric spanners with bounded tree-width would be a strong tool for many geometric applications.

There are spanners with sublinear tree-width. More specifically, since planar graphs have tree-width $\mathcal{O}(\sqrt{n})$ [12] (where n is the number of vertices), for point sets in the Euclidean plane, constant dilation spanners with tree-width $\mathcal{O}(\sqrt{n})$ can be obtained by plane spanner approaches such as the Delaunay triangulation or the greedy triangulation. For point sets in $\mathbb{R}^d$, we point out that the greedy spanner has tree-width $\mathcal{O}(n^{1-1/d})$, using separator results from [11]. While this gives some first results, constructing spanners with tree-width $k \in o(n^{1-1/d})$ will require other constructions. We present a first algorithm that provides a trade-off between tree-width and the dilation, namely an $\mathcal{O}(n/k^{d/(d-1)})$-spanner of tree-width k for sets of points in $\mathbb{R}^d$. We complement our result with a corresponding lower bound, which proves that our algorithm has a worst-case optimal trade-off between tree-width and dilation. We also turn our attention to bounded tree-width spanners with an additional property. The most commonly studied property of geometric spanners is planarity. As a general tool to obtain bounded tree-width plane spanners, we provide a result that establishes a strong dependency between tree-width and the number of edges in sparse connected planar graphs. Using this result and an adapted version of the algorithm provided in [1], we obtain plane spanners with bounded maximum vertex degree as well as bounded tree-width.

2 Bounded tree-width spanners in higher dimensions

In this section, we present an algorithm for constructing bounded tree-width spanners for points in $\mathbb{R}^d$ for constant dimension d.

Algorithm 1 d-`DimensionalBoundedTreeWidthSpanner`(P, k)

```
Require: a set of $n$ points $P \subset \mathbb{R}^d$ and a natural number $k \leq n^{1-1/d}$
Ensure: an $\mathcal{O}(n/k^{d/(d-1)})$-spanner $G = (P, E)$ of tree-width $k$
 1: $\mathcal{EMST} \leftarrow$ Euclidean minimum spanning tree of $P$
 2: if $k = 1$ then
 3:     return $\mathcal{EMST}$
 4: $m \leftarrow \left\lceil \left(k/(30\eta_d c_d 8^d)\right)^{d/(d-1)} + 1 \right\rceil$
 5: Compute a set $\mathcal{T}$ of $m$ disjoint subtrees of $\mathcal{EMST}$, each containing $\mathcal{O}(n/m)$ points.
 6: For $T \in \mathcal{T}$ let $R(T)$ be the vertices in $T$ incident to edges removed by the previous step.
 7: For each $T \in \mathcal{T}$ iterate through its edges $e$ from long to short. If $e$ lies on a path between
    two vertices in $R(T)$, remove the edge. Let $E'(T)$ be the set of remaining edges.
 8: $E' \leftarrow \bigcup_{T \in \mathcal{T}} E'(T)$
 9: $(R, E'') \leftarrow$ greedy $3/2$-spanner $\mathcal{GS}$ for $R = \bigcup_{T \in \mathcal{T}} R(T)$
10: return $G = (P, E' \cup E'')$
```

The subtrees of line 5 are constructed by the removal of separator edges. Removing a separator edge from a tree splits the tree into two subtrees of approximately the same size. For a detailed description of the construction of the subtrees, we refer to the full version of the paper. Using the separator results from [11] we obtain the following result:

▶ **Lemma 2.1.** *Let $P \subset \mathbb{R}^d$ be a set of n points and G the greedy $3/2$-spanner of P. G has tree-width at most $15\eta_d c_d 8^d \cdot n^{1-1/d}$, where η_d is the packing constant in d-dimensional Euclidean space and $c_d \leq 2^{\mathcal{O}(d)}$ is a constant.*

We call a vertex a *representative* of a subtree $T \in \mathcal{T}$ if it was incident to an edge of the $\mathcal{EMST}$ that was removed in the construction of the subtrees in line 5. The set of representatives of all subtrees has size at most $2m - 2$, since each representative is incident to one of the $m - 1$ edges removed. Thus, $\mathcal{GS}$ is the greedy spanner of at most $2m - 2$ points. By Lemma 2.1 and the choice of the number of subtrees the tree-width of $\mathcal{GS}$ is at most k. After line 7 of the algorithm each of the subtrees of the subtrees $T \in \mathcal{T}$ contains only one representative. Thus, the graph G consists of $\mathcal{GS}$ with a tree attached to each of its vertices. These trees do not increase the tree-width, since every tree contains only one vertex from $\mathcal{GS}$. The tree-width of G therefore is at most k.

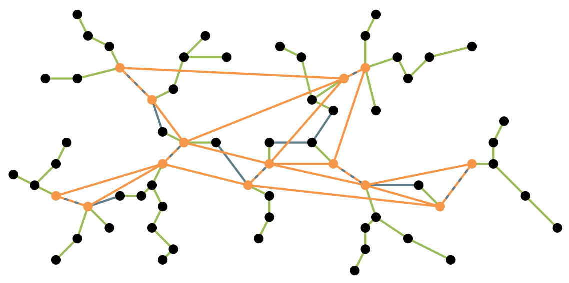


Figure 1 The subtrees (green) of the $\mathcal{EMST}$, the edges removed in line 7 (gray) and the greedy spanner on the representatives (orange).

▶ **Lemma 2.2.** *Given a set $P \subset \mathbb{R}^d$ of n points for some fixed d and a positive integer $k \leq n^{1-1/d}$, Algorithm 1 computes an $\mathcal{O}(n/k^{d/(d-1)})$-spanner for P.*

Proof idea. First ignore that we removed edges in line 7. For two points p, q in the same subtree the dilation is small, since the edges of the path between p and q in the $\mathcal{EMST}$ are short, and since the path contains only few edges. For points in different subtrees the path in the $\mathcal{EMST}$ contains a representative of each of the two subtrees. We can take the paths to these representatives and between the representatives the greedy spanner.

Removing edges in line 7 complicates the argument, but we can use the lengths of the edges removed to bound the distances to the representatives. ◀

The main result of this section now follows:

▶ **Theorem 2.3.** *Given a set $P \subset \mathbb{R}^d$ of n points for some fixed d and a positive integer $k \leq n^{1-1/d}$. There is a geometric spanner of tree-width k on P with a dilation of $\mathcal{O}(n/k^{d/(d-1)})$ and bounded degree that can be computed in time $\mathcal{O}(n^2 \log n)$.*

3 Lower bound

In this section, we show that the bound given in Theorem 2.3 for the dilation of a tree-width bounded spanner on Euclidean point sets is asymptotically tight:

▶ **Theorem 3.1.** *Let $d \geq 2$ be a fixed integer. For positive integers n and $k \leq n^{(d-1)/d} \cdot (5d)^{1/d-2}$, there is a set of n points in $\mathbb{R}^d$, so that every geometric spanner of tree-width k on this set has dilation $\Omega(n/k^{d/(d-1)})$.*

To obtain this result, we construct a set of points resembling a grid. While it is commonly known that a two-dimensional grid has high tree-width, the $\left(k^{1/(d-1)}\right)^d$-grid does not necessarily have tree-width k. We thus construct a set of points resembling the $(h+1)^d$-grid (which is the d-dimensional $(h+1)\times\ldots\times(h+1)$-grid), where $h = \lceil (9d/2 \cdot (k+2))^{1/(d-1)} - 1 \rceil$.

▶ **Lemma 3.2.** *The n^d-grid has tree-width $\geq \frac{2}{9d} \cdot n^{d-1} - 1$.*

Thus, the tree-width of our grid is at least $\frac{2}{9d} \cdot (h+1)^{d-1} - 1 \geq k+1$.

The construction of the grid-like set $P_{d,n,k}$ is as follows: Let $m = \lfloor n/(d \cdot (h+1)^d + d \cdot (h+1)^{d-1}) \rfloor$, be the number of points representing an edge of the grid in $P_{d,n,k}$. For every dimension $i \in \{1, \ldots, d\}$, we define the set

$$P_i = \bigcup_{j_1=0}^{h} \cdots \bigcup_{j_d=0}^{h} \{(j_1 \cdot m, \ldots, j_{i-1} \cdot m, x_i, j_{i+1} \cdot m, \ldots, j_d \cdot m) \mid x_i \in \{0, \ldots, h \cdot m\}\},$$

which consists of $(h+1)^{d-1}$ sets of collinear points representing the grid edges parallel to the axis of the i-th dimension. Each of these collinear sets of points consists of $h \cdot m + 1$ points and represents h edges of the grid. We can now define the set $P_{d,n,k} = \bigcup_{i=1}^{d} P_i$ which represents the whole $(h+1)^d$-grid. (In Figure 2 an example is shown for a 2-dimensional point set.) We call the points $P_{\boxplus} = \{(j_1 \cdot m, \ldots, j_d \cdot m) \mid j_1, \ldots, j_d \in \{0, \ldots, h\}\}$ the *grid points* of the set $P_{d,n,k}$ and two grid points are called *neighbouring* if the Euclidean distance between them is m, which corresponds to them being connected through an edge in the $(h+1)^d$-grid.

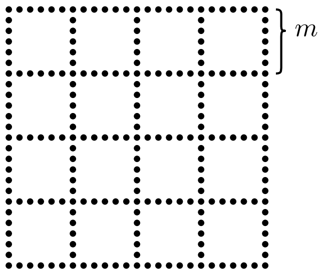


Figure 2 The set of points resembling the $(4+1)^2$-grid where $n = 240$ and $m = 6$.

▶ **Lemma 3.3.** *If G is a geometric $o\left(n/k^{d/(d-1)}\right)$-spanner on $P_{d,n,k}$, then it must be of tree-width $> k$.*

Proof idea. Let G be a $o(n/k^{d/(d-1)})$-spanner on $P_{d,n,k}$. For every pair p, q of neighbouring grid points there is a path $\gamma_{p,q}$ in G connecting them that, because of the dilation bound, roughly follows the points representing the edge in the underlying grid. We can show that these paths must be disjoint outside some certain area around the grid points and can be contracted in a specific way so that contracting one of them does not affected the other paths. Therefore we can look at every pair p, q of neighbouring grid points in some arbitrary order and contract the path $\gamma_{p,q}$ obtaining a graph that has the $(h+1)^d$-grid as a subgraph, proving that G has a $(h+1)^d$-grid minor and therefore must be of tree-width $> k$. ◀

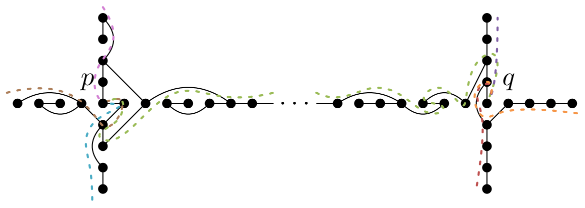


Figure 3 Grid points p and q as well as the path $\gamma_{p,q}$ and paths to the other neighbouring grid points.

4 Minor-3-cores and the tree-width of planar spanners

We have given an asymptotically tight algorithm to obtain spanners of bounded tree-width, with a dilation dependent on the chosen tree-width. We now focus on bounded tree-width spanners with the additional property of being plane.

In general, there are graphs with m edges and tree-width at least $m \cdot \varepsilon$ for a constant ε [7]. We show that this is not true for connected planar graphs, yielding a tight dependency between tree-width and the number of edges:

▶ **Theorem 4.1.** *Let G be a planar connected graph with n vertices and $n + \lfloor (k-1)^2/72 \rfloor$ edges, where $k \geq 2$ is an integer. The tree-width of G is at most k.*

4.1 Minor-3-cores

To obtain this Theorem, we first introduce the concept of *minor-3-cores*, in reference to the k-core which was established by Seidman [17] for the study of social networks.

▶ **Definition 4.2** (Minor-3-Core)**.** Let G be a graph and H a minor of G with minimum degree 3. If there is no minor of minimum degree 3 that has more edges than H i.e., H is an edge maximal minor with this property, we call it a *minor-3-core* of G.

We will speak of *the* minor-3-core instead of a minor-3-core meaning the canonical minor-3-core. For its definition, we refer to the full version of the paper. A useful property of the minor-3-core is that it preserves the tree-width of the original graph.

▶ **Lemma 4.3.** *Given a graph of tree-width $k \geq 3$, the tree-width of the minor-3-core of G is k as well.*

To bound the tree-width of graphs through the size of their minor-3-core we need the following lemma.

▶ **Lemma 4.4.** *Let G be a connected graph with n vertices and $n-1+m$ edges, where $m \geq 1$ is some integer. The minor-3-core of G has at most $2 \cdot (m-1)$ vertices.*

This lemma is the last part we need to prove an upper bound for the tree-width of connected planar graphs.

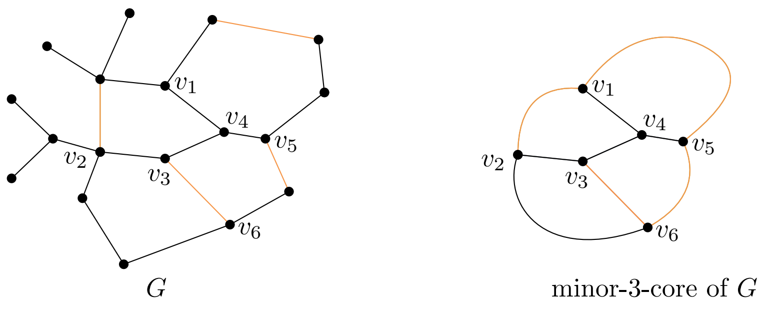


Figure 4 Minor-3-core of a graph G that consists of a tree with 4 additional edges (orange).

Since G is a planar connected graph with $n + \lfloor (k-1)^2/72 \rfloor$ edges, its minor-3-core must be planar and by Lemma 4.4 have at most $\lfloor (k-1)^2/36 \rfloor$ vertices. The tree-width of a planar graph with n vertices by [16] is at most $6\sqrt{n} + 1$. The tree-width of the minor-3-core of G therefore is $6\sqrt{\lfloor (k-1)^2/36 \rfloor} + 1 \leq k$. By Lemma 4.3 we can now conclude that the tree-width of G is either ≤ 2 or exactly k, so in both cases at most k.

4.2 Tree-width of planar spanners

Theorem 4.1 allows us a stronger result than Theorem 2.3 in the Euclidean plane: Adjusting the algorithm given in [1], we obtain a plane spanner with bounded tree-width and degree:

▶ **Corollary 4.5.** *Given a set $P \subset \mathbb{R}^2$ of n points and some positive integer $k \leq 12\sqrt{n-3}$, a plane spanner with tree-width k, maximum vertex degree 4 and dilation $\mathcal{O}(n/k^2)$ can be constructed in $\mathcal{O}(n \log n)$ time.*

Our main adjustment to the algorithm is to choose a plane constant-dilation spanner $\mathcal{S}$ of P with maximum degree 4, as provided in [9], instead of the Delaunay triangulation and to utilise a minimum spanning tree of $\mathcal{S}$ instead of the Euclidean MST. Since the constructed spanner then is a subgraph of $\mathcal{S}$, it is plane and has maximum vertex degree 4. By Theorem 4.1, its tree-width can then be bounded by the given k. Further, the dilation is not affected, which follows directly by Lemma 4 in [1].

References

1 Boris Aronov, Mark de Berg, Otfried Cheong, Joachim Gudmundsson, Herman Haverkort, Michiel Smid, and Antoine Vigneron. Sparse geometric graphs with small dilation. *Computational Geometry*, 40(3):207–219, 2008. doi:10.1016/J.COMGEO.2007.07.004.

2 Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and Ljubomir Perković. Plane spanners of maximum degree six. In *Automata, Languages and Programming: 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I 37*, pages 19–30. Springer, 2010. doi:10.1007/978-3-642-14165-2_3.

3 Prosenjit Bose, Paz Carmi, Mohammad Farshi, Anil Maheshwari, and Michiel Smid. Computing the greedy spanner in near-quadratic time. *Algorithmica*, 58(3):711–729, 2010. doi:10.1007/s00453-009-9293-4.

4 Prosenjit Bose and Michiel Smid. On plane geometric spanners: A survey and open problems. *Comput. Geom. Theory Appl.*, 46(7):818–830, 2013. doi:10.1016/j.comgeo.2013.04.002.

5 Paul Chew. There is a planar graph almost as good as the complete graph. In *Proc. 2nd Symposium on Computational Geometry*, pages 169–177, 1986. doi:10.1145/10515.10534.

6 Gautam Das and Paul J Heffernan. Constructing degree-3 spanners with other sparseness properties. *International Journal of Foundations of Computer Science*, 7(02):121–135, 1996. doi:10.1007/3-540-57568-5_230.

7 Martin Grohe and Dániel Marx. On tree width, bramble size, and expansion. *Journal of Combinatorial Theory, Series B*, 99(1):218–228, 2009. doi:10.1016/j.jctb.2008.06.004.

8 Joachim Gudmundsson and Christian Knauer. Dilation and detours in geometric networks. In *Handbook of Approximation Algorithms and Metaheuristics*, pages 53–69. Chapman and Hall/CRC, 2018.

9 Iyad Kanj, Ljubomir Perkovic, and Duru Turkoglu. Degree four plane spanners: Simpler and better. *J. Comput. Geom.*, 8(2):3–31, 2017. doi:10.20382/jocg.v8i2a2.

10 Rolf Klein and Martin Kutz. Computing geometric minimum-dilation graphs is NP-hard. In *Graph Drawing: 14th International Symposium, GD 2006, Karlsruhe, Germany, September 18-20, 2006. Revised Papers 14*, pages 196–207. Springer, 2007. doi:10.1007/978-3-540-70904-6_20.

11 Hung Le and Cuong Than. Greedy spanners in euclidean spaces admit sublinear separators. *ACM Transactions on Algorithms*, 20(3):1–30, 2024. doi:10.1145/3590771.

12 Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979. doi:10.1137/0136016.

13 Giri Narasimhan and Michiel Smid. *Geometric spanner networks.* Cambridge University Press, 2007. doi:10.1017/cbo9780511546884.

14 Neil Robertson and Paul D. Seymour. Graph minors I. Excluding a forest. *Journal of Combinatorial Theory, Series B*, 35:39–61, 1983. doi:10.1016/0095-8956(83)90079-5.

15 Neil Robertson and Paul D. Seymour. Graph minors II. Algorithmic aspects of tree width. *Journal of Algorithms*, 7:309–322, 1986. doi:10.1016/0196-6774(86)90023-4.

16 Neil Robertson, Paul D. Seymour, and Robin Thomas. Quickly excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 62(2):323–348, 1994. `doi:10.1006/JCTB.1994.1073`.

17 Stephen B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269–287, 1983.

18 Michiel H.M. Smid. The well-separated pair decomposition and its applications. *Handbook of approximation algorithms and metaheuristics*, 13, 2007. `doi:10.1201/9781420010749.ch53`.

The Geodesic Fréchet Distance Between Two Curves Bounding a Simple Polygon

Thijs van der Horst[1,2], Marc van Kreveld[1], Tim Ophelders[*1,2], and Bettina Speckmann[2]

1 **Department of Information and Computing Sciences, Utrecht University, the Netherlands**
`{t.w.j.vanderhorst|m.j.vankreveld|t.a.e.ophelders}@uu.nl`

2 **Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands**
`b.speckmann@tue.nl`

Abstract

We consider a special case of the Fréchet distance between two polygonal curves, where the curves bound a simple polygon and distances are measured via geodesics inside this simple polygon. We significantly improve upon the existing 2-approximation algorithm of Efrat *et al.* (2002). Namely, we present a $(1+\varepsilon)$-approximation algorithm, for any $\varepsilon > 0$, that runs in $\mathcal{O}(\frac{1}{\varepsilon}(n+m\log n)\log nm\log\frac{1}{\varepsilon})$ time for curves with n and m vertices, respectively. To do so, we show how to compute the reachability of specific groups of points in the free space at once and in near-linear time, by interpreting the free space as one between separated one-dimensional curves. Bringmann and Künnemann (2015) previously solved the decision version of the Fréchet distance in this setting in $\mathcal{O}((n+m)\log nm)$ time. We strengthen their result and compute the Fréchet distance between two separated one-dimensional curves in linear time. Finally, we give a linear time exact algorithm if the two curves bound a convex polygon.

Related Version Full version: `arXiv:2501.03834`

1 Introduction

The Fréchet distance is a well-studied similarity measure for curves in a metric space. Most results so far concern the Fréchet distance between two polygonal curves R and B in $\mathbb{R}^d$ with n and m vertices, respectively. The Fréchet distance between two such curves can be computed in $\tilde{\mathcal{O}}(nm)$ time (see e.g. [1, 5]). There is a (nearly) matching conditional lower bound: If the Fréchet distance between polygonal curves can be computed in $\mathcal{O}(n^{2-\varepsilon})$ time for the case $m = n$, then the Strong Exponential Time Hypothesis (SETH) fails [3]. This lower bound even in one dimension holds for any approximation factor less than three [6].

In fact, so far there is no algorithm for general curves that gives any constant-factor approximation in strongly-subquadratic time. Van der Horst *et al.* [12] were the first to present an algorithm that results in an arbitrarily small polynomial approximation factor (n^ε for any $\varepsilon \in (0, 1]$) in strongly-subquadratic time ($\tilde{\mathcal{O}}(n^{2-\varepsilon})$). However, the polynomial approximation barrier is yet to be broken in the general case.

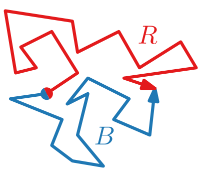
For certain families of "realistic" curves, the SETH lower bound does not apply. For example, when the curves are *c-packed*, Bringmann and Künnemann [4] give a $(1+\varepsilon)$-approximation algorithm, for any $\varepsilon > 0$, that runs in $\tilde{\mathcal{O}}(cn/\sqrt{\varepsilon})$ time. When the curves are *κ-bounded* or *ϕ-low density*, for constant κ or ϕ, Driemel *et al.* [9] give strongly-subquadratic $(1+\varepsilon)$-approximation algorithms as well. Moreover, if the input curves have an imbalanced

[*] Partially supported by the Dutch Research Council (NWO) under project no. VI.Veni.212.260.

number of vertices, then the Fréchet distance of one-dimensional curves can be computed in strongly-subquadratic time without making extra assumptions about the shape of the curves. This was recently established by Blank and Driemel [2], who give an $\tilde{\mathcal{O}}(n^{2\alpha} + n)$-time algorithm when $m = n^\alpha$ for some $\alpha \in (0, 1)$.

In this paper we investigate the Fréchet distance in the presence of obstacles. If the two polygonal curves R and B lie inside a simple polygon P with k vertices and we measure distances by the geodesic distance inside P, then neither the upper nor the conditional lower bound change in a fundamental way. Specifically, Cook and Wenk [8] show how to compute the Fréchet distance in this setting in $\mathcal{O}(k + N^2 \log kN \log N)$ time, with $N = \max\{n, m\}$. For more general polygonal obstacles, Chambers *et al.* [7] give an algorithm that computes the *homotopic* Fréchet distance in $\mathcal{O}(N^9 \log N)$ time, where $N = m + n + k$ is the total number of vertices on the curves and obstacles.

We are investigating the specific setting where the two curves *bound* a simple region, that is, both R and B are simple, meet only at their first and last endpoints, and lie on the boundary of the region. We measure distance by the geodesic Fréchet distance inside that region. If R and B bound a triangulated topological disk D with k faces, then Har-Peled *et al.* [11] give an $\mathcal{O}(\log n)$-approximation algorithm that runs in $\mathcal{O}(k^6 \log k)$ time. If the region is a simple polyon P (see figure) then the SETH lower bound does not apply. Efrat *et al.* [10] give an $\mathcal{O}((n + m) \log nm)$-time 2-approximation algorithm in this setting. In this paper we significantly improve upon their result. In the following we first introduce some notation and then describe our contributions in detail.



Preliminaries. A (polygonal) *curve* R is a piecewise linear function that connects a sequence $r_1, \dots, r_n$ of points, which we refer to as *vertices*. If the vertices lie in the plane, then we say R is two-dimensional[1] and equal to the function $R\colon [1, n] \to \mathbb{R}^2$ where $R(i+t) = (1-t)r_i + tr_{i+1}$ for $i \in \{1, \dots, n-1\}$ and $t \in [0, 1]$. A one-dimensional curve is defined analogously. We assume R is parameterized such that $R(i)$ indexes vertex r_i for all integers $i \in [1, n]$. We denote by $R[x_1, x_2]$ the subcurve of R over the domain $[x_1, x_2]$, and abuse notation slightly to let $R[r, r']$ to also denote this subcurve when $r = R(x_1)$ and $r' = R(x_2)$. Let $R\colon [1, n] \to \mathbb{R}^2$ and $B\colon [1, m] \to \mathbb{R}^2$ be two simple, interior-disjoint curves with $R(1) = B(1)$ and $R(n) = B(m)$. The two curves bound a simple polygon P.

A *reparameterization* of $[1, n]$ is a non-decreasing surjection $f\colon [0, 1] \to [1, n]$. Two reparameterizations f and g of $[1, n]$ and $[1, m]$, describe a *matching* (f, g) between two curves R and B with n and m vertices, respectively, where any point $R(f(t))$ is matched to $B(g(t))$. The matching (f, g) is said to have *cost*

$$\max_t \ d(R(f(t)), B(g(t))),$$

where $d(\cdot, \cdot)$ is the geodesic distance between points in P. A matching with cost at most δ is called a *δ-matching*. The (continuous) *geodesic Fréchet distance* $d_{\mathrm{F}}(R, B)$ between R and B is the minimum cost over all matchings. The corresponding matching is a *Fréchet matching*.

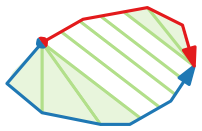
The *parameter space* of R and B is the axis-aligned rectangle $[1, n] \times [1, m]$. Any point (x, y) in the parameter space corresponds to the pair of points $R(x)$ and $B(y)$ on the two curves. A point (x, y) in the parameter space is *δ-close* for some $\delta \geq 0$ if $d(R(x), B(y)) \leq \delta$.

[1] Curves are inherently one-dimensional objects. We abuse terminology slightly to refer to the ambient dimension as the dimension of a curve.

The δ-*free space* $\mathcal{F}_\delta(R, B)$ of R and B is the subset of $[1, n] \times [1, m]$ containing all δ-close points. A point $q = (x', y') \in \mathcal{F}_\delta(R, B)$ is δ-*reachable* from a point $p = (x, y)$ if $x \leq x'$ and $y \leq y'$, and there exists a bimonotone (i.e., monotone in both coordinates) path in $\mathcal{F}_\delta(R, B)$ from p to q. Alt and Godau [1] observe that there is a one-to-one correspondence between δ-matchings between $R[x, x']$ and $B[y, y']$, and bimonotone paths from p to q through $\mathcal{F}_\delta(R, B)$. We abuse terminology slightly and refer to such paths as δ-matchings.

Organization and results. In this paper, we significantly improve upon the result of Efrat *et al.* [10]: we present a $(1+\varepsilon)$-approximation algorithm, for any $\varepsilon > 0$, that runs in $\mathcal{O}(\frac{1}{\varepsilon}(n + m \log n) \log nm \log \frac{1}{\varepsilon})$ time when R and B bound a simple polygon. This algorithm relies on an interesting connection between matchings and nearest neighbors and is described in Section 2. There we also explain how to transform the decision problem for *far* points on B (those who are not the nearest neighbors of any point on R) into a problem between separated one-dimensional curves. Bringmann and Künnemann (2015) previously solved the decision version of the Fréchet distance in this setting in $\mathcal{O}((n + m) \log nm)$ time. In Section 3 we strengthen their result and compute the Fréchet distance between two separated one-dimensional curves in linear time.

Finally, when P is a convex polygon we describe a simple linear-time algorithm (see the full version of this paper). In this setting, we show that a Fréchet matching with a specific structure exists (see figure). We compute the orientation of the parallel part from up to $O(n + m)$ different tangents, which we find using "rotating calipers".



▶ **Theorem 1.** *Let* $R\colon [1, n] \to \mathbb{R}^2$ *and* $B\colon [1, m] \to \mathbb{R}^2$ *be two simple curves bounding a convex polygon, with* $R(1) = B(1)$ *and* $R(n) = B(m)$*. We can construct a Fréchet matching between* R *and* B *in* $\mathcal{O}(n + m)$ *time.*

2 Approximate geodesic Fréchet distance

In this section we describe our approximation algorithm. Let $\varepsilon > 0$ be a parameter. Our algorithm computes a $(1 + \varepsilon)$-approximation to the geodesic Fréchet distance $d_F(R, B)$. It makes use of a $(1 + \varepsilon)$-*approximate decision algorithm.* Given an additional parameter $\delta \geq 0$, the decision algorithm reports either that $d_F(R, B) \leq (1 + \varepsilon)\delta$ or that $d_F(R, B) > \delta$. If $\delta < d_F(R, B) \leq (1 + \varepsilon)\delta$, we may report either answer. In our setting of the problem, we show that the geodesic Fréchet distance is approximately the *geodesic Hausdorff distance* between R and B, that is, at most three times as large. The geodesic Hausdorff distance $\overline{\delta}$ can be computed in $\mathcal{O}((n + m) \log nm)$ time [8, Theorem 7.1]. We then perform binary search over the values $\overline{\delta}, (1 + \varepsilon)\overline{\delta}, \ldots, 3\overline{\delta}$ and apply our approximate decision algorithm at each step. Our decision algorithm runs in $\mathcal{O}(\frac{1}{\varepsilon}(n + m \log n) \log nm)$ time, leading to the following theorem:

▶ **Theorem 2.** *Let* $R\colon [1, n] \to \mathbb{R}^2$ *and* $B\colon [1, m] \to \mathbb{R}^2$ *be two simple curves bounding a simple polygon, with* $R(1) = B(1)$ *and* $R(n) = B(m)$*. Let* $\varepsilon > 0$ *be a parameter. We can compute a* $(1 + \varepsilon)$*-approximation to* $d_F(R, B)$ *in* $\mathcal{O}(\frac{1}{\varepsilon}(n + m \log n) \log nm \log \frac{1}{\varepsilon})$ *time.*

In the remainder of this section we focus on our approximate decision algorithm. At its heart lies a useful connection between matchings and nearest neighbors: for a point r on R its nearest neighbors are the points on B closest to it. Any δ-matching must match each nearest neighbor b of r relatively close to r. Specifically, we prove that b must be matched to a point r' for which all the entire subcurve of R between r and r' are within distance δ

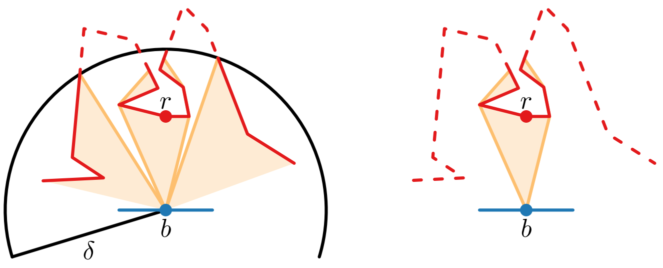


Figure 1 (left) Points r and b with $b \in NN(r)$. The non-dashed red subcurves of R are within geodesic distance δ of b. (right) The (r, b, δ)-nearest neighbor fan.

of b. We capture this relation using (r, b, δ)-*nearest neighbor fans*, illustrated in Figure 1. A nearest neighbor fan $F_{r,b}(\delta)$ corresponds to the point b and the maximal subcurve $R[x, x']$ that contains r and is within geodesic distance δ of b; it is the union of geodesics between b and points on $R[x, x']$.

As r moves monotonically along R, so do its nearest neighbors b along B, together with their nearest neighbor fan $F_{r,b}(\delta)$. While r moves continuously along R, the points b and their fans might jump discontinuously. We show in the full version of this paper how to use the matchings from the fans to efficiently answer the decision question for those points that are part of the fans. See Figure 2 (left) for an illustration of the connection between the δ-free space and nearest neighbor fans.

For sufficiently large values of δ, which includes $\delta = \overline{\delta}$, every point on R is part of a nearest neighbor fan. We distinguish the points on B based on whether they are a nearest neighbor of a point on R. The points that are a nearest neighbor are called *near points*, and others are called *far points*. On B, the near points are part of a nearest neighbor fan, but the far points are not. Far points pose the greatest technical challenge for our algorithm; as can be seen from the structure of the δ-free space around these points (see Figure 2 (right)).

In the full version of this paper, we show how to construct a δ-matching for far points in an approximate manner. Specifically, let $b, b' \in B$ be two points that are involved in nearest neighbor fans, but all points strictly between b and b' are not, that is, they are far points.

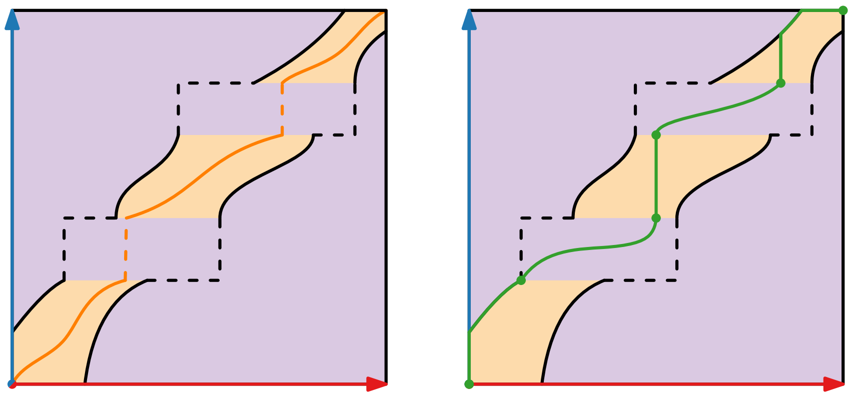


Figure 2 (left) The (r, b, δ)-nearest neighbor fans correspond to a bimonotone region in the δ-free space. The middle partly-dashed curve indicates the nearest neighbor(s) on B of points on R. (right) There is a matching that moves vertically upwards whenever possible. In the dashed rectangles, B has only far points, and the matching becomes more complex.

In this case, there is a point $r \in R$ for which both b and b' are nearest neighbors. Hence $d(b, b') \leq 2d_{\mathrm{F}}(R, B)$. In other words, the geodesic from b to b' is short and separates R from the subcurve $B[b, b']$.

We are going to use this separating geodesic to transform the decision problem for far points into $K = \mathcal{O}(1/\varepsilon)$ one-dimensional problems. Specifically, we discretize the separator with K points, which we call *anchors*, and ensure that consecutive ones have distance at most $\varepsilon\delta$ between them. We snap our geodesics to these anchors, which incurs a small approximation error. Based on which anchor point a geodesic snaps to, we partition the parameter space of R and B' into regions, one for each anchor point. For each anchor point, the lengths of these geodesics snapped to it can be described as the distances between points on two separated one-dimensional curves; this is exactly the one-dimensional problem we now need to solve exactly. We describe our algorithm for handling the one-dimensional problem efficiently in Section 3.

3 Separated one-dimensional curves and propagating reachability

We consider the following problem: Let $\bar{R}$ and $\bar{B}$ be two one-dimensional curves with n and m vertices, respectively, where $\bar{R}$ lies left of the point 0 and $\bar{B}$ right of it. We are given a set $S \subseteq \mathcal{F}_\delta(\bar{R}, \bar{B})$ of $\mathcal{O}(n + m)$ "entrances," for some $\delta \geq 0$. Also, we are given a set $E \subseteq \mathcal{F}_\delta(\bar{R}, \bar{B})$ of $\mathcal{O}(n + m)$ "potential exits." We wish to compute the subset of potential exits that are δ-reachable from an entrance. We call this procedure *propagating reachability information* from S to E. See Figure 3 for an illustration.

The problem of propagating δ-reachability information has already been studied by Bringmann and Künnemann [4]. In case S lies on the left and bottom sides of the parameter space and E lies on the top and right sides, they give an $\mathcal{O}((n+m) \log nm)$ time algorithm. We are interested in a more general case however, where S and E may lie anywhere in the parameter space. We make heavy use of the concept of *prefix-minima* to develop an algorithm for our more general setting that has the same running time as the one described by Bringmann and Künnemann [4] (see Theorem 4). Furthermore, our algorithm is able

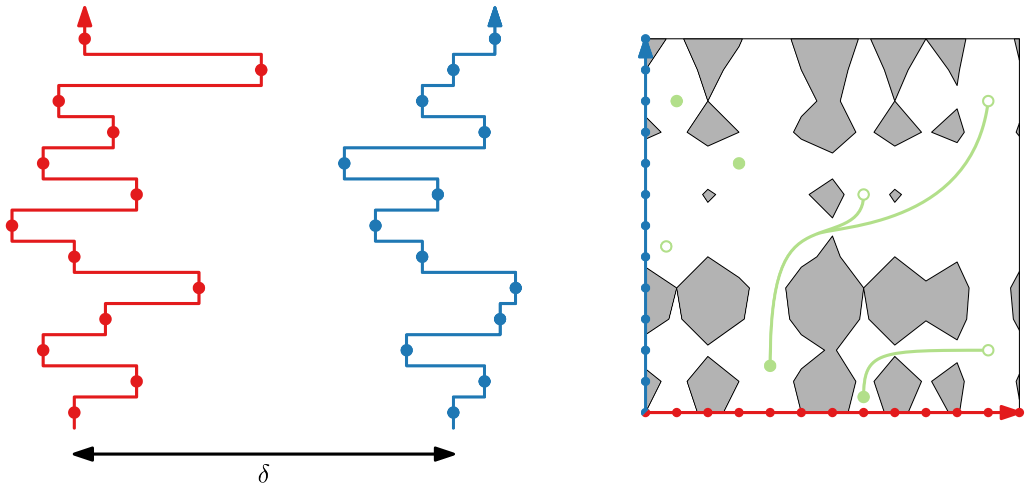


Figure 3 (left) A pair of separated, one-dimensional curves $\bar{R}$ and $\bar{B}$, drawn stretched vertically for clarity. (right) The free space $\mathcal{F}_\delta(\bar{R}, \bar{B})$ corresponding to the matching, with a set of entrances (disks) and potential exits (circles). Some matchings between entrances and exits are drawn.

to actually compute a Fréchet matching between $\bar{R}$ and $\bar{B}$ in linear time (see Theorem 3), while Bringmann and Künnemann require near-linear time for only the decision version.

▶ **Theorem 3.** *Let $\bar{R}$ and $\bar{B}$ be two separated one-dimensional curves with n and m vertices. We can construct a Fréchet matching between $\bar{R}$ and $\bar{B}$ in $\mathcal{O}(n+m)$ time.*

▶ **Theorem 4.** *Let $\bar{R}$ and $\bar{B}$ be two separated one-dimensional curves with n and m vertices, where no two vertices coincide. Let $\delta \geq 0$, and let $S \subseteq \mathcal{F}_\delta(\bar{R}, \bar{B})$ and $E \subseteq \mathcal{F}_\delta(\bar{R}, \bar{B})$ be sets of $\mathcal{O}(n+m)$ points. We can compute the set of all points in E that are δ-reachable from a point in S in $\mathcal{O}((n+m)\log nm)$ time.*

References

1 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5:75–91, 1995. doi:10.1142/S0218195995000064.

2 Lotte Blank and Anne Driemel. A faster algorithm for the Fréchet distance in 1d for the imbalanced case. In *proc. 32nd Annual European Symposium on Algorithms (ESA)*, volume 308 of *LIPIcs*, pages 28:1–28:15, 2024. doi:10.4230/LIPICS.ESA.2024.28.

3 Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *proc. 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 661–670, 2014. doi:10.1109/FOCS.2014.76.

4 Karl Bringmann and Marvin Künnemann. Improved approximation for Fréchet distance on c-packed curves matching conditional lower bounds. *International Journal of Computational Geometry & Applications*, 27(1-2):85–120, 2017. doi:10.1142/S0218195917600056.

5 Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. Four soviets walk the dog: Improved bounds for computing the Fréchet distance. *Discrete & Computational Geometry*, 58(1):180–216, 2017. doi:10.1007/s00454-017-9878-7.

6 Kevin Buchin, Tim Ophelders, and Bettina Speckmann. SETH says: Weak Fréchet distance is faster, but only if it is continuous and in one dimension. In *proc. 30th Annual Symposium on Discrete Algorithms (SODA)*, pages 2887–2901, 2019. doi:10.1137/1.9781611975482.179.

7 Erin W. Chambers, Éric Colin de Verdière, Jeff Erickson, Sylvain Lazard, Francis Lazarus, and Shripad Thite. Homotopic Fréchet distance between curves or, walking your dog in the woods in polynomial time. *Computational Geometry*, 43(3):295–311, 2010. doi:10.1016/J.COMGEO.2009.02.008.

8 Atlas F. Cook IV and Carola Wenk. Geodesic Fréchet distance inside a simple polygon. *ACM Transactions on Algorithms*, 7(1):9:1–9:19, 2010. doi:10.1145/1868237.1868247.

9 Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the Fréchet distance for realistic curves in near linear time. *Discrete & Computational Geometry*, 48(1):94–127, 2012. doi:10.1007/s00454-012-9402-z.

10 Alon Efrat, Leonidas J. Guibas, Sariel Har-Peled, Joseph S. B. Mitchell, and T. M. Murali. New similarity measures between polylines with applications to morphing and polygon sweeping. *Discrete & Computational Geometry*, 28(4):535–569, 2002. doi:10.1007/S00454-002-2886-1.

11 Sariel Har-Peled, Amir Nayyeri, Mohammad R. Salavatipour, and Anastasios Sidiropoulos. How to walk your dog in the mountains with no magic leash. *Discrete & Computational Geometry*, 55(1):39–73, 2016. doi:10.1007/S00454-015-9737-3.

12 Thijs van der Horst, Marc J. van Kreveld, Tim Ophelders, and Bettina Speckmann. A subquadratic n^{ε}-approximation for the continuous Fréchet distance. In *proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1759–1776, 2023. doi:10.1137/1.9781611977554.ch67.

Sliding Squares in Parallel[*]

Hugo A. Akitaya[1], Sándor P. Fekete[2], Peter Kramer[2], Saba Molaei[3], Christian Rieck[4], Frederick Stock[1], and Tobias Wallner[2]

1 **Miner School of Computer and Information Sciences, University of Massachusetts Lowell, MA, USA**
`hugo_akitaya@uml.edu`, `frederick_stock@student.uml.edu`

2 **Department of Computer Science, TU Braunschweig, Braunschweig, Germany**
`s.fekete@tu-bs.de`, `{kramer, wallner}@ibr.cs.tu-bs.de`

3 **Sharif University of Technology, Tehran, Iran**
`molaeisaba1@gmail.com`

4 **Department of Discrete Mathematics, University of Kassel, Kassel, Germany**
`christian.rieck@mathematik.uni-kassel.de`

Abstract

We consider algorithmic problems motivated by modular robotic reconfiguration, for which we are given n unlabeled square-shaped modules in a starting configuration and need to find a schedule of sliding moves to transform it into a desired goal configuration, maintaining connectivity at all times.

Recent work has aimed at minimizing the total number of moves, resulting in fully sequential schedules that can perform reconfigurations in $\mathcal{O}(n^2)$ moves, or $\mathcal{O}(nP)$ for an arrangement with bounding box of perimeter P. We provide results in the sliding square model that exploit parallel robot motion, resulting in an optimal speedup to achieve reconfiguration in worst-case optimal makespan of $\mathcal{O}(P)$. We also show a tight bound on the complexity of the problem by showing that even deciding the possibility of reconfiguration within makespan 1 is NP-complete.

Related Version `10.48550/arXiv.2412.05523`

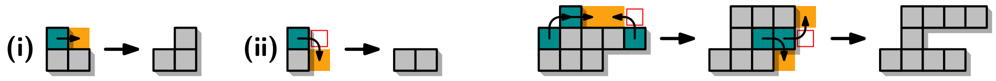
1 Introduction

Reconfiguring an arrangement of objects is a fundamental problem in both theory and practice, often in a setting with strong geometric flavor. A typical task arises from relocating a (potentially large) collection of agents from a given start into a desired goal configuration in an efficient manner, while avoiding collisions between objects or violating other constraints, such as maintaining connectivity of the overall arrangement.

In recent years, the problem of modular robot reconfiguration [7, 18, 19] has enjoyed particular attention [1, 2, 3, 4, 6, 15] in the context of Computational Geometry: In the *sliding square model* introduced by Fitch, Butler, and Rus [13], a given start configuration of n modules, each occupying a square grid cell, must be transformed by a sequence of atomic, sequential moves (shown in Figure 1(a)) into a target arrangement, without losing connectivity of the underlying grid graph. Aiming at minimizing the total number of moves, the mentioned previous work has resulted in considerable progress, recently establishing universal configuration in $\mathcal{O}(nP)$ for a 2-dimensional arrangement of n modules with bounding box perimeter size P [4], and $\mathcal{O}(n^2)$ in three dimensions [1, 15]. This work on sequential reconfiguration differ from practical settings, in which modules can exploit parallel motion to achieve much faster reconfiguration times—which is also a more challenging objective, as it requires coordinating the overall motion plan to maintain connectivity and avoid collisions.

[*] This work was partially supported by the German Research Foundation (DFG), project "Space Ants", FE 407/22-1, by NSF grant CCF-2348067, and by DFG grant 522790373.

Our work focuses on parallel reconfiguration, allowing modules to perform moves simultaneously with the objective of minimizing the total time until completion, the *makespan*. For a detailed overview of related literature and recent results, we refer to the full version [5].



(a) Two types of move can be performed by individual modules: **(i)** Slides and **(ii)** convex transitions.

(b) Some moves can be performed in parallel transformations, as shown here.

Figure 1 Our model allows for two types of moves to be chained into collision-free transformations. In this paper, we show the symmetric difference of a transformation using turquoise and yellow.

Our contributions. We provide the following tight results for *parallel* reconfiguration in the sliding square model. For technical details, we refer to the full version of our paper [5].

1. We prove that the *unlabeled* version of parallel reconfiguration is NP-complete, even when trying to decide the existence of a schedule with the smallest possible makespan of 1.
2. We give a weakly in-place algorithm that achieves makespan $\mathcal{O}(P)$, where P is the perimeter of the union of the bounding boxes of start and target configurations.

Preliminaries. We study reconfiguration in the *parallel sliding square* model as follows. An *instance* $\mathcal{I}$ is composed of an initial configuration C_1 of n robotic *modules* that must be reconfigured into a target configuration C_2. In any configuration, each square module occupies a unique cell of the infinite integer grid. We navigate the grid using cardinal directions (north, south, east, west) and cell adjacency. To reconfigure C_1 into C_2, we employ schedules of atomic, parallel *transformations*. A transformation selects a subset of modules to move, each performing either a *slide* or a *convex transition* to travel into an adjacent cell, see Figure 1(a). Note that in our figures, the path denoting a convex transition is shown containing a circular arc for clarity. Such path would be accurate if the corners of modules were rounded. In our model, both take an identical (unit) duration to complete, which allows us to extend the existing notion of move counting in sequential models [4, 9, 16, 10] to transformation counting for parallel moves. The *makespan* of a reconfiguration schedule thus corresponds to the number of transformations.

During each transformation, a connectivity-preserving *backbone* must be maintained. To this end, we call a set M of modules in a configuration C *free* if $C \setminus M'$ is a valid configuration for any $M' \subseteq M$; a transformation $C_1 \rightarrow C_2$ is legal exactly if the moving modules are free in C_1 and C_2. Furthermore, a transformation is collision-free exactly if all modules can move along their designated path at a constant rate, without overlapping with any other module in the process. We identify three types of collisions that must be avoided, see Figure 2.

(i) Any two moves collide if their target cells are identical, or if they constitute a swap.
(ii) Two convex transitions collide if they share an intermediate cell (highlighted in red).
(iii) Two moves collide if their paths meet orthogonally at endpoints.

In the full version of our paper, we discuss these in depth and show that some collisions of type **(iii)** can be avoided with only a constant overhead in the number of transformations. Our collision model is less restrictive than some models of previous work on parallel reconfiguration [10, 14], which require the motion paths of parallel moves to be pairwise disjoint. On the other hand, it is more constrained than [11] which does not forbid collisions of type **(iii)**. (However, note that [11] does not impose the single backbone condition.)

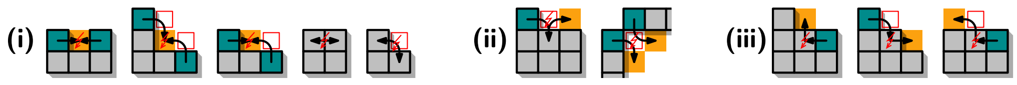


Figure 2 Examples of all collision types: Modules cannot share or swap cells, convex transitions cannot share an intermediate cell, and moves with connected paths cannot meet orthogonally.

Consider an instance $\mathcal{I} = (C_1, C_2)$. Throughout this paper, we denote the bounding boxes of C_1 and C_2 by B_1 and B_2 , respectively. As in existing literature, we assume that B_1 and B_2 share a south-west corner and call a schedule for $\mathcal{I}$ *in-place* exactly if no intermediate configuration exceeds the union $B_1 \cup B_2$ by more than one module [16, 4]. For technical reasons, we desire bounding box dimensions that are multiples of three, as well as a three-wide empty column at the eastern boundary. We define the *extended bounding box* B' from B by expanding it in each of the cardinal directions by at most three units so that the dimensions of B' are multiples of three (see Figure 5). We say that a schedule is *weakly in-place* if and only if intermediate configurations are restricted to the union of the bounding boxes extended by a constant amount, e.g., $B'_1 \cup B'_2$. We refer to the standard definition as *strictly in-place*.

2 Computational complexity

In this section, we sketch a reduction from Planar Monotone 3Sat [8] showing that Parallel Sliding Squares is NP-complete. In particular, we prove the following theorem.

▶ **Theorem 2.1.** *Let $\mathcal{I}$ be an instance of* Parallel Sliding Squares*. It is* NP*-complete to decide whether there exists a feasible schedule of makespan* 1 *for $\mathcal{I}$.*

An example of the overall construction is depicted in Figure 3. The involved *variable* and *clause gadgets* are marked in yellow and blue, respectively. While the gadgets mostly retain their shape in the target configuration (visualized as gray modules), a single module must be moved per variable gadget. There are two feasible chain moves for this, each representing either a positive or negative Boolean value assignment, as shown in Figure 3.

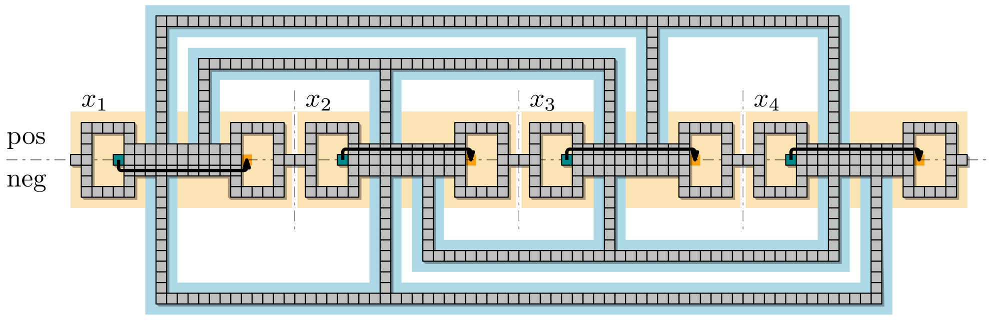


Figure 3 Our construction for $\varphi = (x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_4})$. The depicted transformation represents the satisfying assignment $\alpha(\varphi) = (\texttt{true}, \texttt{false}, \texttt{false}, \texttt{false})$.

Proof sketch. Due to the way in which both configurations are connected, the transformation depicted in Figure 3 disconnects each variable gadget from either all its positive or negative clauses. Thus, a satisfying assignment maps to a feasible schedule. As these chain moves are unique schedules of makespan 1, we also easily obtain the opposite direction. ◀

It is easy to solve our reduction instances in makespan 2, implying the following.

▶ **Corollary 2.2.** *Unless* P = NP*, there is no polynomial-time* $(1+\delta)$*-approximation algorithm for* Parallel Sliding Squares *with* $\delta \in [0, 1)$.

Our result highlights a complexity gap between the parallel sliding square model and closely related models for parallel transformation. In particular, Fekete et al. [11, 12] studied a related model in which makespan 2 is NP-hard to decide, but makespan 1 is in P.

3 A worst-case optimal algorithm

We now introduce a four-phase, polynomial-time algorithm for Parallel Sliding Squares. The high-level goal of our algorithm is to transform the entire configuration into *meta-modules*, small units of modules that cooperate to provide greater reconfiguration capability, which we then exploit. Unlike our work, the existence of such meta-modules is a common constraint on the input in related models [6, 11, 12, 14, 17] due to their high capability.

A meta-module in our work has a distinct *center cell* v surrounded by eight modules, such that all vertex-adjacent cells to v are full; the cell v itself may be empty or occupied.

Figure 4 Meta-modules are either clean or solid, depending on whether the center cell is occupied.

Our approach computes reconfiguration schedules linear in the perimeter P_1 and P_2 of the bounding boxes B_1 and B_2 of C_1 and C_2, respectively. This is asymptotically worst-case optimal. Instances that induce a matching lower bound can easily be constructed, e.g., an instance asking for a large "L" to be rotated into "⅂" requires at least $\Omega(P_1 + P_2)$ moves.

▶ **Theorem 3.1.** *For any instance* $\mathcal{I}$ *of* Parallel Sliding Squares*, we can compute a feasible, weakly in-place schedule of* $\mathcal{O}(P_1 + P_2)$ *transformations in polynomial time.*

Our algorithm consists of four phases: In **Phase (I)**, we identify a subconfiguration used as a "backbone" (similar to [14]), and gather $\Theta(P_1)$ many modules around a piece of this backbone, thereby enhancing its connectivity (as in [4]). We use the flexibility of this piece to construct a sweep-line structure out of meta-modules in **Phase (II)**, that is then used in **Phase (III)** to efficiently compact and transform the remaining configuration into meta-modules forming an xy-monotone histogram, similar to [1]. In **Phase (IV)**, this histogram can then be transformed into any other such histogram with the same number of modules, effectively morphing between a "start histogram" and "target histogram", using similar techniques as in [11]. To reach our target configuration, we then simply apply **Phases (I-III)** in reverse. We refer to Figure 5 for a visualization of the different phases of the algorithm. Although several of our techniques are inspired by previous work, they differ substantially in our context of parallel reconfiguration and considerable changes were needed.

Phase (I): Gathering squares. We use an underlying connected substructure, referred to as a *skeleton*, from the initial configuration to guide the reconfiguration. This tree-like skeleton functions as a backbone around which we move modules toward a "root" module h, making a subtree of the skeleton "thick" (i.e., the neighborhood of every cell in this part of the skeleton is occupied). We call this thick subskeleton an *exoskeleton*, denoted by X_h. The exoskeleton has a higher connectivity than the skeleton, making it easier to transform into a sweep line.

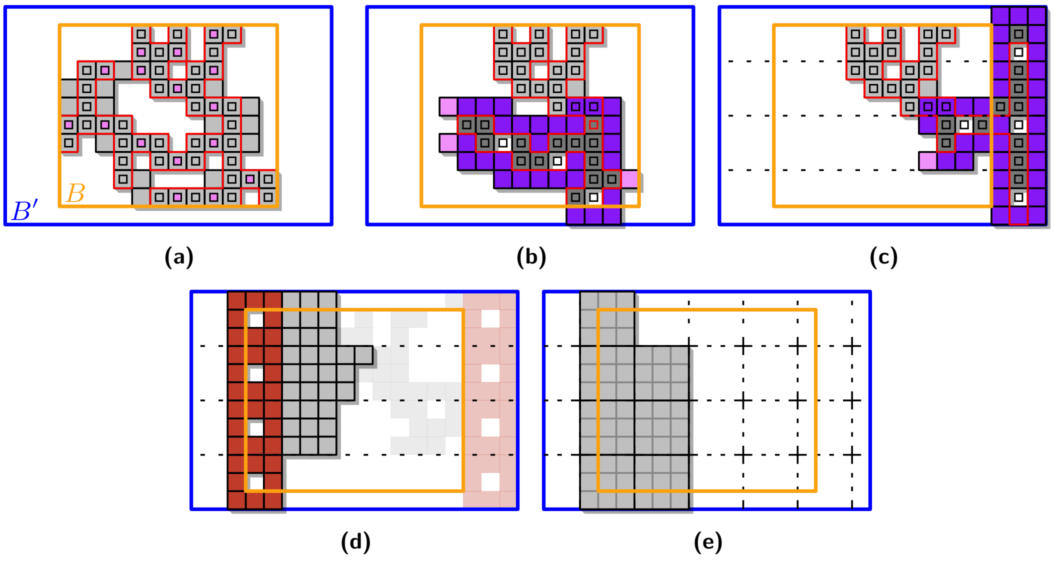


Figure 5 The high-level overview of our approach: We show (a) an initial configuration C and its skeleton (red border), (b) the gathered exoskeleton (in dark gray and purple) and (c) its resulting scaffold, (d) the sweep line (in red) in its initial and final state, and finally (e) the xy-monotone histogram of meta-modules.

A similar idea is also used by Hurtado et al. [14]. However, we achieve a stronger result in the classic sliding model via careful definition of the skeleton and movement. Upon completion of the gathering process, the exoskeleton X_h contains $\Theta(P)$ modules.

Phase (II): Scaffolding. Once the exoskeleton X_h is sufficiently large, we reconfigure it to be "T-shaped" and contain the east edge of the extended bounding box. Note that the interior (*core*) cells of the exoskeleton X_h are not necessarily all occupied; connectivity is instead maintained through their neighborhood. We proceed in three steps:

1. First "compact" X_h so that its core contains no empty cells.
2. Then choose an easternmost node c in the core of X_h and grow a horizontal path from c eastward (Figures 6(a) to 6(c)), until a 3×3 square is formed outside the bounding box.
3. Grow the path north until it reaches the corner of the bounding box, then grow a path south until it reaches the adjacent corner (Figures 6(d) to 6(f)).

Phase (III): Sweeping into a histogram. Once **Phase (II)** concludes, we are left with an intermediate configuration that contains a highly regular "scaffold" configuration with a wall of modules at the easternmost edge, as shown in Figure 6(f). Using only local operations, we transform this vertical section into meta-modules, forming our "sweep line" ℓ. We use local protocols to move the sweep line across the extended bounding box from east to west, consuming and pushing modules ahead of it, as shown in Figures 7(a) to 7(d). These consumed modules are pushed ahead of the sweep line, up to the western edge of the bounding box, at which point they start trailing behind in a "compacted" form, see Figures 7(a) to 7(c). This compacted form yields a number of histograms attached to ℓ, see Figure 7(f).

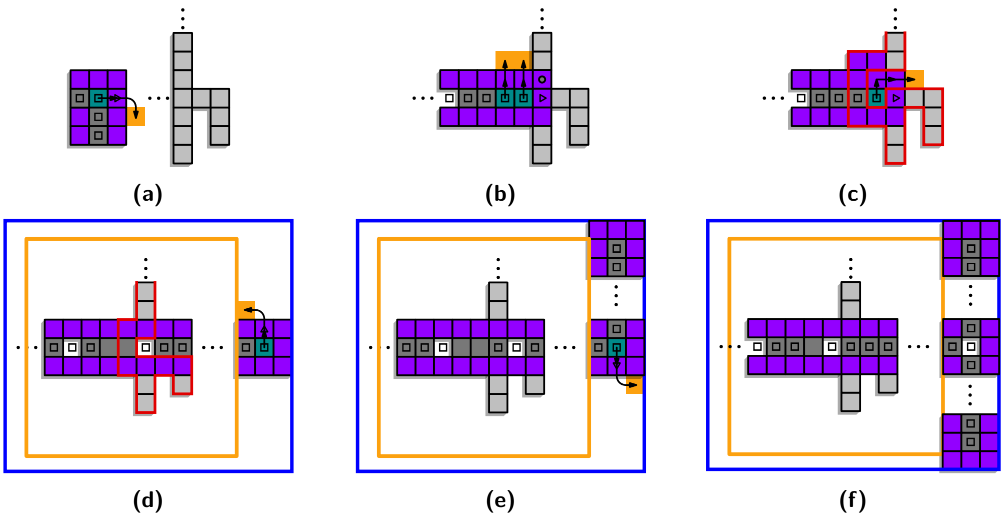


Figure 6 Building the scaffolding. The extended bounding box is shown in blue.

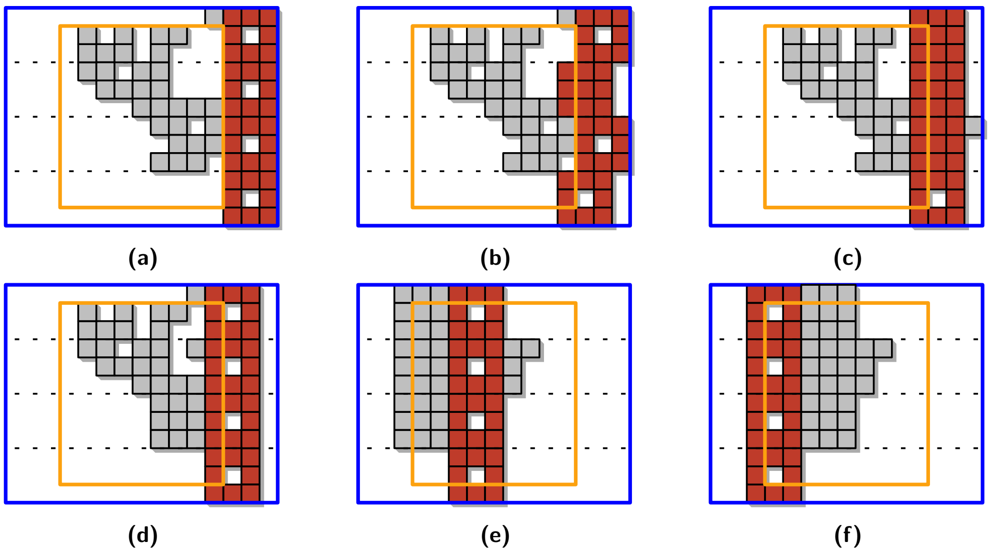


Figure 7 Moving the sweep line (red modules). The extended bounding box is shown in blue.

Phase (IV): Histograms of meta-modules. Once we complete the sweep, i.e., **Phase (III)**, we have a configuration that has a sweep line at the western boundary, with a compact configuration of modules east of the sweep line. This configuration resembles an x-monotone histogram of meta-modules, with at most a constant number of modules at each y-coordinate not belonging to a meta-module. We group the modules into meta-modules that form a histogram aligned with a regular grid, creating a *scaled histogram*, as visualized in Figure 8.

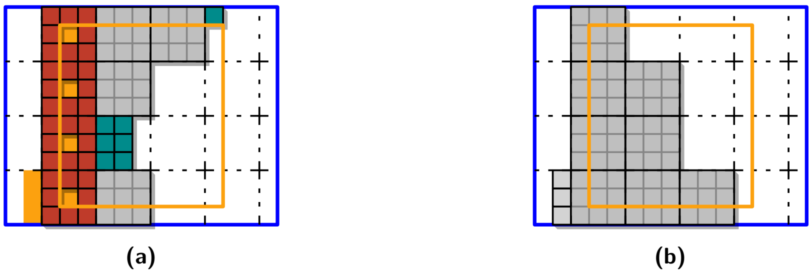


Figure 8 After the completing the sweep, we construct xy-monotone histogram of meta-modules.

Any two such histograms can be efficiently reconfigured into one another by a schedule that never places any module outside their union, i.e., a strictly in-place schedule. To realize this, we employ a simple sweep-line algorithm similar to [11, 12] that incrementally reduces the symmetric difference between the start and target histograms by mapping between excess occupied and unoccupied cells using two diagonal bisectors. Once we have obtained the scaled target histogram, we can begin applying the processes described in the above phases in reverse, eventually obtaining the target configuration.

Each phase takes $\mathcal{O}(P_1)$ transformations, so its reverse takes $\mathcal{O}(P_2)$. We therefore obtain an overall makespan of $\mathcal{O}(P_1 + P_2)$; a detailed analysis can be found in the full version.

4 Conclusions and future work

We have provided a number of new results for reconfiguration in the sliding squares model, making full use of parallel robot motion. While these outcomes are worst-case optimal, there are still a number of possible generalizations and extensions.

We can generalize our hardness results for the labeled setting to show that deciding whether makespan 2 is possible is NP-complete. Furthermore, we provide an efficient approach to decide if a schedule of makespan 1 exists. We can adapt our algorithmic results to this setting by "sorting" the modules in the xy-monotone configuration in $\mathcal{O}(P)$ transformations.

Previous work has progressed from two dimensions to three. Can our approach be extended to higher dimensions? We are optimistic that significant speedup can be achieved, but the intricacies of three-dimensional topology may require additional tools.

References

1 Zachary Abel, Hugo A. Akitaya, Scott Duke Kominers, Matias Korman, and Frederick Stock. A universal in-place reconfiguration algorithm for sliding cube-shaped robots in a quadratic number of moves. In *Symposium on Computational Geometry (SoCG)*, pages 1:1–1:14, 2024. doi:10.4230/LIPIcs.SoCG.2024.1.

2 Hugo A. Akitaya, Esther M. Arkin, Mirela Damian, Erik D. Demaine, Vida Dujmovic, Robin Y. Flatland, Matias Korman, Belén Palop, Irene Parada, André van Renssen, and Vera

Sacristán. Universal reconfiguration of facet-connected modular robots by pivots: The $\mathcal{O}(1)$ musketeers. *Algorithmica*, 83(5):1316–1351, 2021. doi:10.1007/s00453-020-00784-6.

3 Hugo A. Akitaya, Erik D. Demaine, Andrei Gonczi, Dylan H. Hendrickson, Adam Hesterberg, Matias Korman, Oliver Korten, Jayson Lynch, Irene Parada, and Vera Sacristán. Characterizing universal reconfigurability of modular pivoting robots. In *Symposium on Computational Geometry (SoCG)*, pages 10:1–10:20, 2021. doi:10.4230/LIPIcs.SoCG.2021.10.

4 Hugo A. Akitaya, Erik D. Demaine, Matias Korman, Irina Kostitsyna, Irene Parada, Willem Sonke, Bettina Speckmann, Ryuhei Uehara, and Jules Wulms. Compacting squares: Input-sensitive in-place reconfiguration of sliding squares. In *Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 4:1–4:19, 2022. doi:10.4230/LIPIcs.SWAT.2022.4.

5 Hugo A. Akitaya, Sándor P. Fekete, Peter Kramer, Saba Molaei, Christian Rieck, Frederick Stock, and Tobias Wallner. Sliding squares in parallel, 2024. doi:10.48550/arXiv.2412.05523.

6 Greg Aloupis, Sébastien Collette, Mirela Damian, Erik D. Demaine, Robin Flatland, Stefan Langerman, Joseph O'Rourke, Suneeta Ramaswami, Vera Sacristán, and Stefanie Wuhrer. Linear reconfiguration of cube-style modular robots. *Computational Geometry*, 42(6-7):652–663, 2009. doi:10.1016/J.COMGEO.2008.11.003.

7 Zack Butler and Daniela Rus. Distributed planning and control for modular robots with unit-compressible modules. *The International Journal of Robotics Research*, 22(9):699–715, 2003. doi:10.1177/02783649030229002.

8 Mark de Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *International Journal on Computational Geometry and Applications*, 22(3):187–206, 2012. doi:10.1142/S0218195912500045.

9 Adrian Dumitrescu and János Pach. Pushing squares around. *Graphs and Combinatorics*, 22(1):37–50, 2006. doi:10.1007/s00373-005-0640-1.

10 Adrian Dumitrescu, Ichiro Suzuki, and Masafumi Yamashita. Motion planning for metamorphic systems: feasibility, decidability, and distributed reconfiguration. *Transactions on Robotics*, 20(3):409–418, 2004. doi:10.1109/TRA.2004.824936.

11 Sándor P. Fekete, Phillip Keldenich, Ramin Kosfeld, Christian Rieck, and Christian Scheffer. Connected coordinated motion planning with bounded stretch. *Autonomous Agents and Multi-Agent Systems*, 37(2):43, 2023. doi:10.1007/S10458-023-09626-5.

12 Sándor P. Fekete, Peter Kramer, Christian Rieck, Christian Scheffer, and Arne Schmidt. Efficiently reconfiguring a connected swarm of labeled robots. *Autonomous Agents and Multi-Agent Systems*, 38(2):39, 2024. doi:10.1007/S10458-024-09668-3.

13 Robert Fitch, Zack J. Butler, and Daniela Rus. Reconfiguration planning for heterogeneous self-reconfiguring robots. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2460–2467, 2003. doi:10.1109/IROS.2003.1249239.

14 Ferran Hurtado, Enrique Molina, Suneeta Ramaswami, and Vera Sacristán Adinolfi. Distributed reconfiguration of 2d lattice-based modular robotic systems. *Autonomous Robots*, 38(4):383–413, 2015. doi:10.1007/S10514-015-9421-8.

15 Irina Kostitsyna, Tim Ophelders, Irene Parada, Tom Peters, Willem Sonke, and Bettina Speckmann. Optimal in-place compaction of sliding cubes. In *Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 31:1–31:14, 2024. doi:10.4230/LIPICS.SWAT.2024.31.

16 Joel Moreno and Vera Sacristán. Reconfiguring sliding squares in-place by flooding. In *European Workshop on Computational Geometry (EuroCG)*, pages 32:1–32:7, 2020. URL: https://www1.pub.informatik.uni-wuerzburg.de/eurocg2020/data/uploads/papers/eurocg20_paper_32.pdf.

17 Irene Parada, Vera Sacristán, and Rodrigo I. Silveira. A new meta-module design for efficient reconfiguration of modular robots. *Autonomous Robots*, 45(4):457–472, 2021. `doi:10.1007/S10514-021-09977-6`.

18 Daniela Rus and Marsette Vona. Crystalline robots: Self-reconfiguration with compressible unit modules. *Autonomous Robots*, 10:107–124, 2001. `doi:10.1109/MRA.2007.339623`.

19 Serguei Vassilvitskii, Mark Yim, and John Suh. A complete, local and parallel reconfiguration algorithm for cube style modular robots. In *International Conference on Robotics and Automation (ICRA)*, pages 117–122, 2002. `doi:10.1109/ROBOT.2002.1013348`.

Multi-Covering a Point Set by m Disks with Minimum Total Area

Aaron T. Becker[1], **Sándor P. Fekete**[2], **Mariem Guitouni**[1], **Chek-Manh Loi**[2], **and Michael Perk**[2]

1 **University of Houston**
`atbecker@uh.edu,mguitoun@CougarNet.uh.edu`

2 **Department of Computer Science, TU Braunschweig**
`s.fekete@tu-bs.de, {loi, perk}@ibr.cs.tu-bs.de`

Abstract

In general multi-coverage, we are given a set of n points $p_1, \dots, p_n$ in the plane. The task is to choose m locations $q_1, \dots, q_m$ and assign a radius r_j to each q_j, such that each p_i is covered by disks centered at $\kappa(p_i) \geq 1$ different q_j with corresponding radius r_j, such that the sum of disk areas is minimized. We provide fast heuristics and exact methods that compute provably optimal solutions, which we extend to the generalization in which disk centers are subject to separation constraints.

Related Version arXiv:2502.13773

1 Introduction

Covering a set of geometric locations is an important optimization problem that arises in different areas. As shown in Figure 1a, this includes scenarios from robotics, e.g., controlling a set of ground locations from a finite set of drones with downward communication links [15, 7], requiring a set of different altitudes that balance safe separation between drones with reliable communication to the ground. The latter requires sufficient signal strength, so communication areas (and thus energy consumption) depend quadratically on the altitude. For any location, observation with more than one drone is often needed to ensure sufficiently robust coverage. Similar problems exist in diverse application domains, including wireless sensor networks [1, 3, 4, 17, 5], facility placement [2, 6] and pesticide application [14, 8].

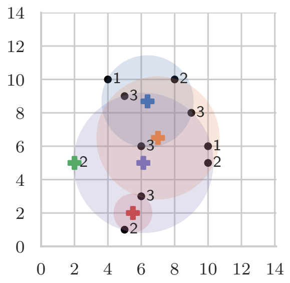
In the *general multi-coverage* (GMC) problem, we are given a point set S, m sensors, and a coverage function $\kappa : S \to \mathbb{N}$; the goal is to assign a center q_i and radius r_i for each disk $i \in [1, \dots, m]$ so that each $p_i \in S$ is covered by at least $\kappa(p_i)$ disks. The objective is to minimize the sum of disk areas $\pi \sum_{i=1}^{m} r_i^2$. An additional constraint arises by enforcing sufficient separation between coverage centers: For distance ℓ, the *dispersive multi-coverage* problem (DGMC) asks for a GMC with $\|q_i - q_j\| \geq \ell$ for all $i \neq j$ and $i, j \in [1, \dots, m]$. (See Figures 1b and 1c for examples of optimal solutions.)

2 Related Work

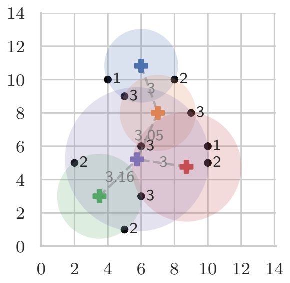
Alt et al. [2] studied the GMC with $\kappa(p) = 1$. A related, but simpler, problem explored in previous works uses a given set C of disk centers; with given coverage multiplicities $\kappa(p_i)$, this is known as the non-uniform minimum-cost multi-cover (MCMC) problem. If $\forall p \in S, \kappa(p) = k$ it is referred to as the uniform MCMC.

Approximation algorithms with constant factors depending on κ for uniform and non-uniform MCMC were given by Abu-Affash et al. [1] and BarYehuda and Rawitz [3], respectively. Bhowmick et al. [6] achieved constant approximation for non-uniform MCMC independent of κ. Huang et al. [12, 13] gave a PTAS for $\kappa(S) > 1$.

(a) Kilobot robots



(b) GMC



(c) DGMC

Figure 1 (a) Ground-based Kilobot robots, commanded by overhead controllers via infrared communication [16]. Optimal (minimum total area) solutions with $n = 10$, $m = 5$, and κ between 1 and 3. (b) Without separation constraints. (c) Enforcing a minimum distance of $\ell = 3$.

Also related is placing a minimum number of unit disks to multi-cover a point set of size $|S| = n$. Gao et al. [10] gave a 5-approximation with runtime $\mathcal{O}(n + \kappa_{\max})$, and a 4-approximation algorithm with runtime $\mathcal{O}(n^2)$. Filipov and Tomova studied coverage with the minimum number of unit disks [9], providing a stochastic algorithm with expected complexity $\mathcal{O}(n^2)$.

3 Solving GMC: Lower Bounds

We consider approaches for solving the GMC problem, implying lower bounds for the DGMC.

3.1 GMC heuristic

The heuristic starts with an initial k-means solution to partition the point set into m clusters. These clusters are expanded to ensure each point p is covered $\kappa(p)$ times and locally optimized to minimize the total area of the disks. See full version for a detailed description.

3.2 Integer Programming

To formulate the GMC as an IP, we need a discrete set of candidate sensor positions. We discuss computing a (preferably small) sufficient set C of candidate disks in Section 3.2.1. Given C, we can formulate the integer program in Section 3.2.2.

3.2.1 Computing the Candidate Set

Without separation constraints, there are three ways that a set of points $S' \subseteq S$ can be covered optimally (i.e., with minimum-area) by a disk.

a) For $S' = \{p_1\}$, a disk centered at p_1 with radius 0 is optimal.
b) For $S' = \{p_1, p_2\}$, there is a unique disk with radius $\frac{\|p_i - p_j\|}{2}$ centered at the midpoint of $\overline{p_1, p_2}$ that is the minimum-area disk that contains both points.
c) For $|S'| \geq 3$, any disk covering S' can be shrunk until it has either (i) two points $p_1, p_2 \in S'$ on its boundary (see case b) or (ii) at least three points $p_1, p_2, p_3 \in S'$ on its boundary.

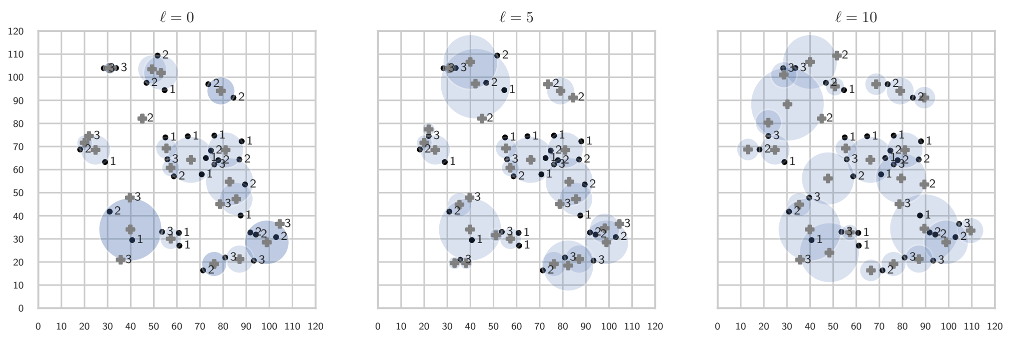


Figure 2 Solutions from *uni_lg* with $n = 30$ and $m = 30$ and different separation constraints.

The above allows us to enumerate all necessary disks for the GMC: We add a disk with radius 0 and center p_i for all points $p_i \in S$. Then for all pairs $p_i, p_j \in S$, we compute the disk centered between p_i, p_j. For all triples $p_i, p_j, p_k \in S$, we compute the unique disk that has p_i, p_j and p_k on its boundary. When the triangle between the points is obtuse, a disk in C (defined by two of the points) already contains the third point and has a smaller area. Thus, we only add a disk defined by three points if we encounter an acute triangle.

In total, this yields $\mathcal{O}(n^3)$ possible positions. Using a *k-d-* or ball-tree one can find the set $S' \subseteq S$ of points that intersect a given disk in $\mathcal{O}(\sqrt{n} + |S'|)$ time. This yields a worst-case runtime of $\mathcal{O}(n^4)$ to enumerate all elements of C, but with better performance in practice.

3.2.2 GMC IP Formulation

For every disk d_i in the candidate set C, we define integer variables x_i that encode how often each disk is used in the solution. The constraints ensure that at most m disks are placed and every point $p_j \in S$ is covered by at least $\kappa(p_j)$ disks.

$$\begin{array}{lll} \text{minimize} & \pi \cdot \sum_{d_i \in C} r_i^2 x_i & \\ \text{subject to} & \sum_{d_i \in C} x_i \leq m & \\ & \sum_{\substack{d_i \in C \\ p_j \in d_i}} x_i \geq \kappa(p_j), & \forall p_j \in S \\ & x_i \in \{0, \dots, m\}, & \forall d_i \in C \end{array}$$

4 Upper Bounds: Enforcing separation constraints

While the DGMC can be formulated as a quadratic program with non-convex constraints for disk separation, solving this to optimality is challenging. Thus, we again work with a discretized candidate set C and modify the Integer Programming formulation from Section 3.2.2. However, unlike for the GMC, C does not necessarily contain disks of an optimal solution. It could even be that no selection of disks from C provides any feasible solution to the DGMC. Therefore, we modify C to improve the quality of our solutions; see Section 4.2.

We use the GMC as a lower bound to the DGMC. Comparing this to any DGMC solution with a discretized candidate set allows us to evaluate the quality of the solution for the (non-discretized) DGMC. We found our solutions to be very close to lower bounds provided by the GMC IP; see Section 5.

4.1 Introducing Separation Constraints

We start with the integer program from Section 3.2.2. Separation between two disks is achieved by using binary variables (to ensure that each disk can be selected at most once) and adding the following constraints to prevent two disks with distance $\leq \ell$ (for some distance ℓ) from being selected.

$$x_i + x_j \leq 1 \quad \forall d_i, d_j \in C : \|q_i - q_j\| \leq \ell. \tag{1}$$

For $\mathcal{O}(n^3)$ possible disks, this would yield $\mathcal{O}(n^6)$ possible constraints, which is impractical for interesting instances. Thus, we only add the violating constraints in an iterative fashion.

Due to the separation requirement, we can further add a *clique constraint* for each violating disk that ensures at most one disk is selected within a distance $< \frac{\ell}{2}$. For some disk d_i this clique constraint can be formulated as

$$\sum_{\substack{d_j \in C \\ d(q_i, q_j) < \ell/2}} x_j \leq 1. \tag{2}$$

Equation (2) includes separations from Equation (1), so we add Equation (1) for distant violating disk pairs, and Equation (2) for the $\frac{\ell}{2}$ neighbors of each violating disk. We limit the size of the resulting DGMC IP, by only adding Equation (2) for a disk d_i if no clique was added for some other disk d_j in the clique.

4.2 Modifying the Candidate Set

The next idea is to enhance the candidate set C by promising disks for coverage. In the GMC solution, single outlier points p with $\kappa(p) > 1$ are often covered using $\kappa(p)$ many drones that cover only p. This is no longer possible when ensuring disk separation. For each point p with $\kappa(p) > 1$, we extend the candidate set C by $\kappa(p)$ small disks, that respect the separation constraints, i.e., we construct a regular $\kappa(p)$-gon with side length ℓ centered around p.

To speed up the solver, we can focus on *small* disks. To that end, we check for the largest disk d_i used in the GMC solution and remove all other disks that have a radius that is greater than $\alpha \cdot r_i$. The factor α compromises between the size of C and solution quality.

5 Results

Experiments were carried out on a regular desktop workstation with an AMD Ryzen 9 7900 (12×3.7 GHz) CPU and 88 GB of RAM. Code and data are available[1]. Instances were generated uniformly in a 100 m × 100 m canvas. Values of $\kappa(p)$ for all points were sampled uniformly from $\{1, 2, 3\}$. This yields the instances sets *uni_sm* ($n = 20, 30, \ldots, 200$ and $m = 20$), *uni_lg* ($n = 30, 40, \ldots, 300$ and $m = 30$), and *uni_fix_n* ($m = 5, 10, \ldots, 100$ and $n = 250$). We generated five instances for each parameter combination.

5.1 GMC

We compare the GMC heuristic and the integer program in terms of runtime and total area. GMC IP requires time to set up the solver, i.e., (i) computing the candidate set C which takes $\mathcal{O}(n^4)$ and (ii) building the model which takes $\mathcal{O}(n|C|)$. The solver is executed on the resulting integer program.

[1] `https://gitlab.ibr.cs.tu-bs.de/alg/disc-covering`

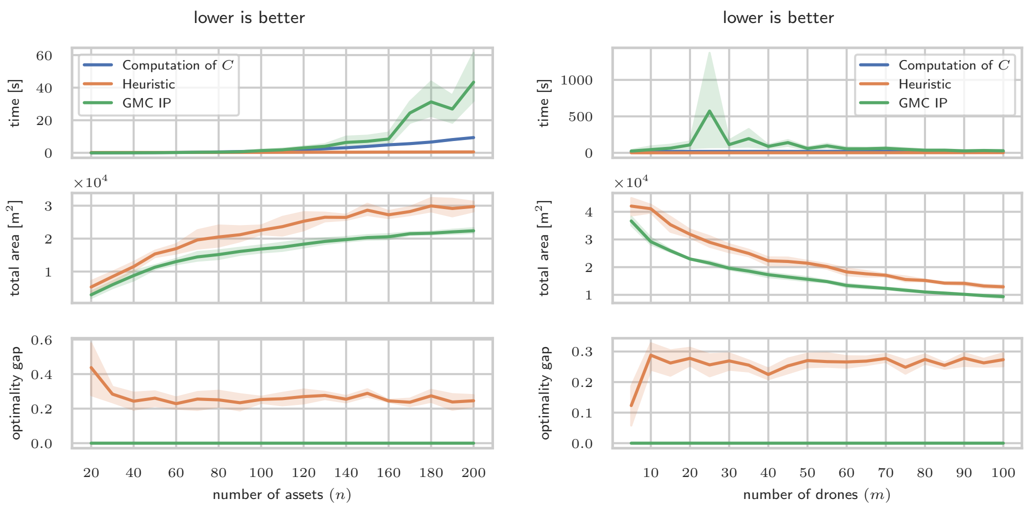


Figure 3 Comparison of runtime, total area, and optimality gap between the GMC IP solver and the heuristic. On all plots, lower is better. (left) *uni_sm*; fixed $m = 20$ variable n. (right) *uni_fix_n*; fixed $n = 250$ variable m.

Figure 3 shows that the runtime of the GMC IP solver is significantly higher than of the GMC heuristic. Computing the candidate set is challenging for larger instances, but we do not observe the worst-case behavior in runtime. For *uni_fix_n*, having m between 20 and 35 the GMC IP needs significantly more time to obtain provable optimal solutions.

The lower row of Figure 3 shows a comparison between GMC heuristic and GMC IP in terms of solution quality. The plot shows the optimality gap that is the relative gap between the found solution versus the optimal solution ($(C_{\text{alg}} - C_{\text{ip}})/C_{\text{alg}}$). For both fixed m and n, the optimality gap remains stable at around 27.5 % in different instances. The only exception being the case where $m \geq n$ in which the iterative algorithm gives slightly worse results.

5.2 DGMC

In Section 4.2 we presented different candidate set strategies that are now compared in terms of solution quality and runtime. For all the experiments, we enable clique constraints and extend the candidate set by small disks.

We ran these exploratory experiments on the benchmark set *uni_sm*, as the workstations ran out of memory for larger instances. After identifying the best parameters, we ran another experiment on the bigger benchmark set *uni_lg*, i.e., for more points and more drones. For all the experiments, we chose a fixed $\ell = 5$. For a fixed value of ℓ, it is more reasonable to variate the number of points n, as more drones increase the difficulty of sparsifying the solution and lead to more infeasibilities.

5.2.1 Parameters of DGMC IP

Figure 4 (left) shows the described tradeoff for the α parameter: α controls the size of the largest disk in the candidate set C. The top row shows that reducing C improves the solver times significantly. At the same time, the bottom row shows how the solution quality decreases with a smaller set. Setting $\alpha = 1.2$ provides excellent tradeoff between solution quality and runtime, reducing runtime while almost maintaining the same solution quality as the original set, i.e., $\alpha = \infty$.

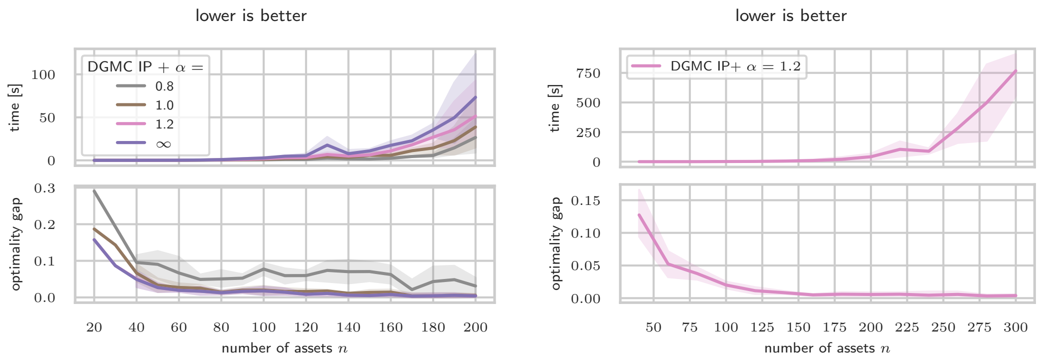


Figure 4 (left) Tradeoff between reducing the candidate set C and the optimality gap for the DGMC IP on *uni_sm*; $m = 20$ and $\ell = 5$. For comparison, we only display instances that were feasible for all α values, removing 14 of the 95 instances. (right) Runtime and optimality gap of the DGMC IP on *uni_lg*; $m = 30$ and $\ell = 5$.

5.2.2 Large benchmark *uni_lg*

First we ran the GMC IP on the benchmark set to obtain the lower bounds for the DGMC. There is a single instance with $n = 300$ that could not be solved within the memory limit; for the remainder of this section, we will exclude this instance.

Based on the results from *uni_sm*, we ran the DGMC IP on the larger benchmark set *uni_lg* with a time limit of 900 s for the solver and set $\alpha = 1.2$. Note that without reducing C (i.e. with $\alpha = \infty$), we cannot reliably solve the larger instances, i.e., instances with more than 250 points, as the integer program requires too much memory.

Figure 4 (right) shows that we can solve all instances close to provable optimality. For smaller instances with $n \leq 100$ the optimality gaps are higher than 2 %. For larger instances, the DGMC IP solver was unable to find optimal solutions for the discretized DGMC (see Section 4) and was terminated due to a timeout. Despite early termination, DGMC IP found solutions with an optimality gap below 0.7 % for all these instances. Note that as we are comparing against the GMC IP solutions (without separation constraints), the gaps to an optimal solution of the DGMC are smaller than what can be seen here.

6 Conclusions

Directions for future work include an extension to covered assets in 3D, which is natural for domains such as flying robots, space applications, or undersea sensor networks. Calculating candidate centers is still possible, but more complicated [11]. The high speed of the iterative approximation may be applicable for dynamic targets, or for adjusting the sensor positions and radii when a sensor is added or deleted. A quadratic program for DGMC is suitable for small problems, and could be initialized with the DGMC IP solutions to speed computation.

References

1 A Karim Abu-Affash, Paz Carmi, Matthew J Katz, and Gila Morgenstern. Multi cover of a polygon minimizing the sum of areas. *International Journal of Computational Geometry & Applications*, 21(06):685–698, 2011.

2 Helmut Alt, Esther M. Arkin, Hervé Brönnimann, Jeff Erickson, Sándor P. Fekete, Christian Knauer, Jonathan Lenchner, Joseph S. B. Mitchell, and Kim Whittlesey. Minimum-cost coverage of point sets by disks. In *Symposium on Computational Geometry (SoCG)*, pages 449–458, 2006. doi:10.1145/1137856.1137922.

3 Reuven Bar-Yehuda and Dror Rawitz. A note on multicovering with disks. *Computational Geometry*, 46(3):394–399, 2013.

4 Francesco Bernardini, Daniel Biediger, Ileana Pineda, Linda Kleist, and Aaron T. Becker. Strongly-connected minimal-cost radio-networks among fixed terminals using mobile relays and avoiding no-transmission zones. In *IEEE International Conference on Automation Science and Engineering*, 2024.

5 Francesco Bernardini, Mohammadreza Shahsavar, Aaron T. Becker, and Julien Leclerc. Joint optimization of target tracking and communication in a shared network. In *2024 IEEE Texas Symposium on Wireless and Microwave Circuits and Systems (WMCS)*, pages 1–6, April 2024. doi:10.1109/WMCS62019.2024.10619030.

6 Santanu Bhowmick, Kasturi Varadarajan, and Shi-Ke Xue. A constant-factor approximation for multi-covering with disks. In *Symposium on Computational Geometry (SoCG)*, pages 243–248, 2013.

7 K. Boudjit and C. Larbes. Detection and implementation autonomous target tracking with a quadrotor ar.drone. In *12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, pages 223–230, 2015.

8 Jahid Chowdhury Choton and Pavithra Prabhakar. Optimal multi-robot coverage path planning for agricultural fields using motion dynamics. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11817–11823, May 2023. doi:10.1109/ICRA48891.2023.10160265.

9 Stefan M. Filipov and Fani N. Tomova. Covering a set of points with a minimum number of equal disks via simulated annealing. In Ivan Georgiev, Maria Datcheva, Krassimir Georgiev, and Geno Nikolov, editors, *Numerical Methods and Applications*, pages 134–145, Cham, 2023. Springer Nature Switzerland.

10 Xuening Gao, Longkun Guo, and Kewen Liao. Fast approximation algorithms for multiple coverage with unit disks. In *Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 185–193. IEEE, 2022.

11 Bernd Gärtner. Fast and robust smallest enclosing balls. In *European symposium on algorithms*, pages 325–338. Springer, 1999.

12 Ziyun Huang, Qilong Feng, Jianxin Wang, and Jinhui Xu. PTAS for minimum cost multi-covering with disks. In *Symposium on Discrete Algorithms (SODA)*, pages 840–859. SIAM, 2021.

13 Ziyun Huang, Qilong Feng, Jianxin Wang, and Jinhui Xu. PTAS for minimum cost multicovering with disks. *SIAM Journal on Computing*, 53(4):1181–1215, 2024.

14 Ratan Lal and Pavithra Prabhakar. Time-optimal multi-quadrotor trajectory planning for pesticide spraying. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7965–7971. IEEE, 2021.

15 Zhihao Liu, Yuanyuan Shang, Timing Li, Guanlin Chen, Yu Wang, Qinghua Hu, and Pengfei Zhu. Robust multi-drone multi-target tracking to resolve target occlusion: A benchmark. *IEEE Transactions on Multimedia*, 25:1462–1476, 2023.

16 Michael Rubenstein, Christian Ahler, and Radhika Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *2012 IEEE international conference on robotics and automation*, pages 3293–3298. IEEE, 2012.

17 Martin Zoula and Jan Faigl. Wireless communication infrastructure building for mobile robot search and inspection missions. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5970–5976, May 2024. doi:10.1109/ICRA57147.2024.10611561.

Exact Algorithms for Minimum Dilation Triangulation

Sándor P. Fekete[1], **Phillip Keldenich**[1], **and Michael Perk**[1]

1 **Department of Computer Science, TU Braunschweig**
`s.fekete@tu-bs.de, {keldenich, perk}@ibr.cs.tu-bs.de`

Abstract

We provide a spectrum of new theoretical insights and practical results for finding a Minimum Dilation Triangulation (MDT), a natural geometric optimization problem of considerable previous attention: Given a set P of n points in the plane, find a triangulation T, such that a shortest Euclidean path in T between any pair of points increases by the smallest possible factor compared to their straight-line distance. No polynomial-time algorithm is known for the problem; moreover, evaluating the objective function involves computing the sum of (possibly many) square roots. On the other hand, the problem is not known to be NP-hard. We provide practically robust methods and implementations for computing the MDT for benchmark sets with up to 30,000 points in reasonable time on commodity hardware, based on new geometric insights into the structure of optimal edge sets. Previous methods only achieved results for up to 200 points, so we extend the range of optimally solvable instances by a factor of 150.

Moreover, we resolve an open problem by establishing a lower bound of 1.44116 on the dilation of the regular 84-gon (and thus for arbitrary point sets), improving the previous worst-case lower bound of 1.4308 and greatly reducing the remaining gap to the upper bound of 1.4482 from the literature. In the process, we provide optimal solutions for regular n-gons up to $n = 100$.

Related Version arXiv:2502.18189

1 Introduction

Triangulating a set of points to optimize some objective is one of the classical problems in computational geometry. On the practical side, it has applications in wireless sensor networks [25, 26], mesh generation [1], computer vision [23], geographic information systems [24] and many other areas [5].

In this paper, we provide new results and insights for a previously studied, natural objective that considers triangulations as sparse structures with relatively low cost for ensuing detours: The *dilation* of a triangulation T of a point set P is the worst-case ratio (among all $s, t \in P$) between the shortest s-t-path $\pi_T(s,t)$ in T and the Euclidean distance $d(s,t)$ between s and t, i.e. $\rho(T) = \max\{{|\pi_T(s,t)|}/{d(s,t)} \mid s,t \in P, s \neq t\}$. The Minimum Dilation Triangulation (MDT) problem asks for a triangulation T that minimizes the dilation $\rho(T)$, see Figure 1 for examples. Despite this importance and attention, actually computing a Minimum Dilation Triangulation is a challenging problem. Its computational complexity is still unresolved, signaling that there may not be a simple and elegant algorithmic solution that scales well.

Our contributions We present practically robust methods and implementations for computing an MDT for benchmark sets with up to 30,000 points in reasonable time on commodity hardware, based on new geometric insights into the structure of optimal edge sets. Previous methods only achieved results for up to 200 points (involving one computational routine of complexity $\Theta(n^4)$ instead of our improved complexity of $O(n^2 \log n)$), so we extend the range of practically solvable instances by a factor of 150. We also resolve an open problem

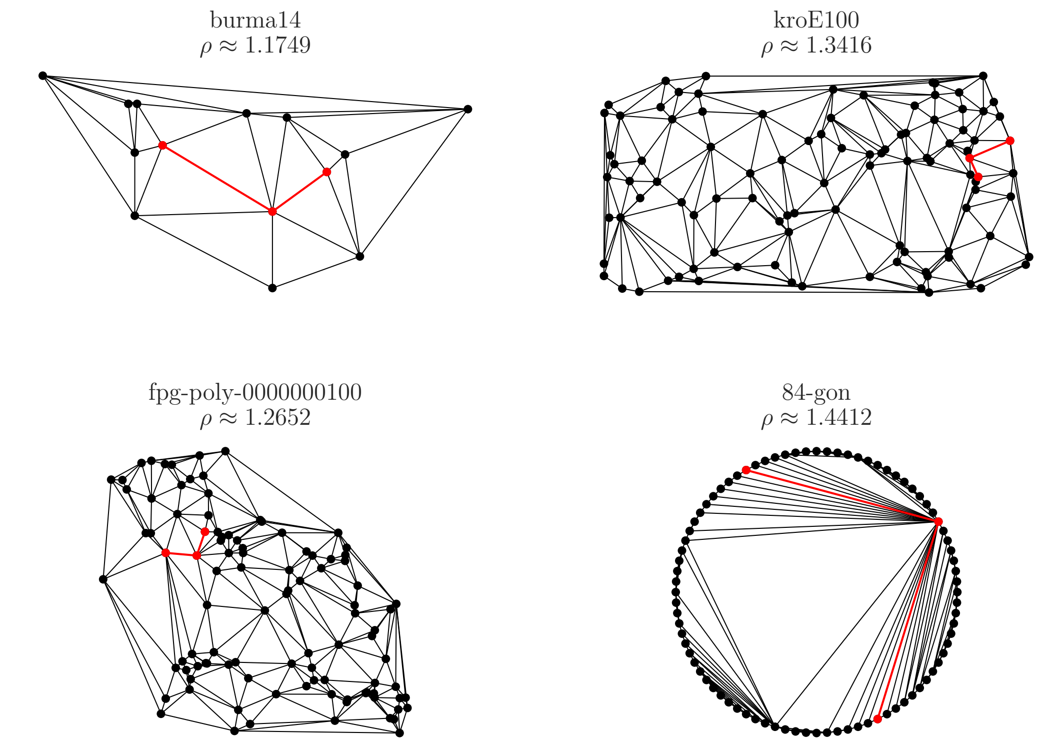


Figure 1 MDT solutions for four instances. The red edges indicate a dilation-defining path.

from Dumitrescu and Ghosh [9] by establishing a lower bound of 1.44116 on the dilation of the regular 84-gon. This improves the previous worst-case lower bound of 1.4308 from the regular 23-gon and greatly reduces the remaining gap to the upper bound of 1.4482 from [22]. In the process, we provide optimal solutions for regular n-gons up to $n = 100$.

Related work The complexity of finding the MDT is unknown [11, 18]. Giannopoulos et al. [12] prove that finding the minimum dilation graph with a limited number of edges is NP-hard. Cheong et al. [4] show that finding a spanning tree of given dilation is also NP-hard. Kozma [16] proves NP-hardness for minimizing the expected distance between random points in a triangulation, with edge weights instead of Euclidean distances. All practical approaches in the literature are based on fixed-precision arithmetic. Klein [14] used an enumeration algorithm to find an optimal MDT for up to 10 points. Dorzán et al. [8] present heuristics for the MDT and evaluate their performance on instances with up to 200 points. Instances with up to 70 points were solved by Brandt et al. [3] using integer linear programming techniques and the edge elimination strategy from Knauer and Mulzer [15]. Recently, Sattari and Izadi [21] presented an exact algorithm based on branch and bound that was evaluated on instances with up to 200 points.

2 Exact algorithms

Now we present two exact algorithms: IncMDT is an incremental method that uses a SAT solver for iterative improvement, until it can prove that no better solution exists. BinMDT is based on a binary search for the optimal dilation ρ; once the lower and upper bound are reasonably close, the approach falls back to IncMDT to reach a provably optimal solution.

2.1 Enumerating possible edges

We implemented a novel and practically efficient scheme for enumerating a set of edges that induces a supergraph of the MDT with dilation *strictly less* than a given bound ρ. The approach is based on the well-known *ellipse property* (used in [3, 12, 15]) and enumerates candidate edges using a quadtree-based filtered incremental nearest-neighbor search that identifies potential neighbors while excluding points in so-called *dead sectors*. Due to space constraints, all details are deferred to the full version.

Our enumeration scheme significantly reduces the number of edges to consider, improving runtime efficiency. It also computes a dilation threshold $\vartheta(st)$ for each edge st to quickly exclude edges when lowering the dilation bound. As part of our computation, we also obtain an initial triangulation and its dilation, as well the intersecting possible edges $I(st)$ for each possible edge st. In both algorithms, we may gradually discover triangulations with lower dilation; these are used to exclude additional edges using the precomputed dilation thresholds $\vartheta(e)$. To keep track of the status of each edge, we insert all points and possible edges into a graph data structure we call *triangulation supergraph*. In this structure, we mark each edge as *possible*, *impossible* or *certain*. Initially, all enumerated edges are *possible*. If, at any point, all edges intersecting an edge e become *impossible*, e becomes *certain*. If an *impossible* edge becomes *certain* or vice versa, the graph does not contain a triangulation any longer. If this happens, we say we encounter an *edge conflict*.

2.2 SAT formulation

Given a triangulation supergraph $G = (P, E)$, we model the problem of finding a triangulation on *possible* and *certain* edges using the following simple SAT formulation. Let $E_p \subseteq E$ be the set of non-*impossible* edges when the SAT formulation is constructed. For each edge $e \in E_p$, we have a variable x_e. We use the following clauses in our formulation.

$$\neg x_{e_1} \vee \neg x_{e_2} \qquad \forall e_1, e_2 \in E_p : e_2 \in I(e_1) \tag{1}$$

$$x_e \vee \bigvee_{e_j \in I(e)} x_{e_j} \qquad \forall e \in E_p \tag{2}$$

Clauses (1) ensure crossing-freeness and clauses (2) ensure maximality. When an edge e becomes certain, we add the clause x_e; similarly, when an edge becomes impossible, we add the clause $\neg x_e$. Both algorithms are based on this simple formulation; in the following, we describe how they use and modify it to find an MDT.

Clause generation The following subproblem, which we call *dilation path separation*, arises in both our algorithms: Given a dilation bound ρ, a triangulation supergraph $G = (P, E)$ excluding only edges that cannot be in any triangulation with dilation less than ρ, a current triangulation T and a pair of points $s, t \in P$ such that $|\pi_T(s, t)| \geq \rho \cdot d(s, t)$, find a clause C that is (a) violated by T and (b) satisfied by any triangulation T' with $\rho(T') < \rho$. For a description of how we compute E' in practice, see the full version.

▶ **Lemma 2.1.** *Assuming a polynomial-time oracle for comparing sums of square roots, there is a polynomial-time algorithm that solves the dilation path separation problem.*

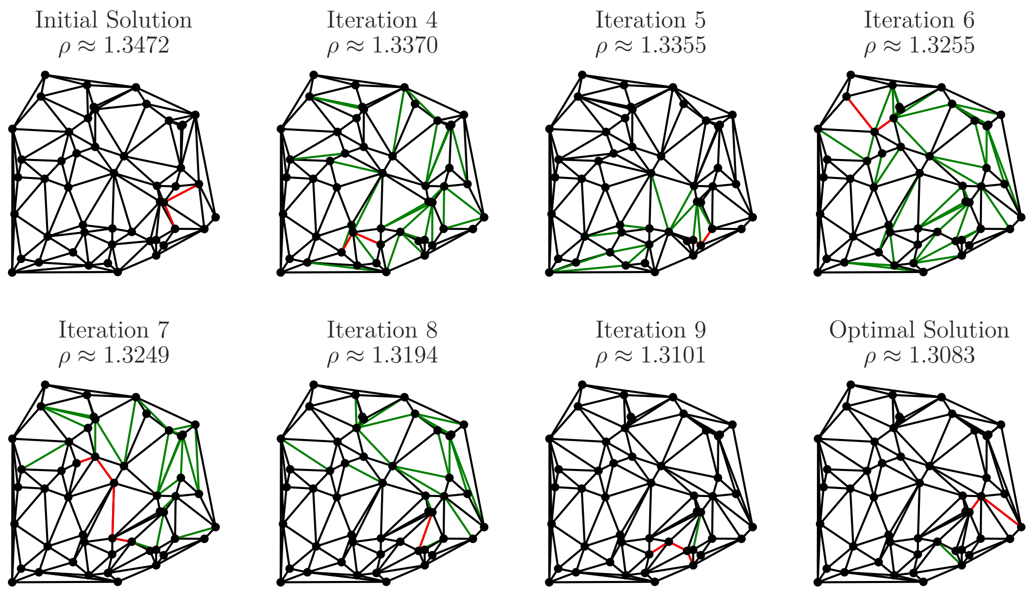


Figure 2 Progress of the incremental algorithm on an instance with $n = 50$ points. Green edges indicate changes in the triangulation, red edges indicate a dilation-defining path.

Proof. Let Π be the set of all s-t-paths π in G with $|\pi| < \rho \cdot d(s,t)$. We begin by observing that, along every path $\pi \in \Pi$, there is an edge $e \in E$ that is not in T; otherwise, we get a contradiction to $|\pi_T(s,t)| \geq \rho \cdot d(s,t)$. Let $E' \subseteq E \setminus T$ be a set of edges such that for each $\pi \in \Pi$, there is an edge $e \in E'$ on π. Then, $C = \bigvee_{e \in E'} x_e$ is a clause that satisfies the requirements; note that if Π is empty, the empty clause can be returned.

T contains no edge from E', so C is violated by T. Furthermore, if a triangulation T' with $\rho(T') < \rho$ does not contain any of the edges in E', it contains none of the paths in Π. Therefore, $\pi_{T'}(s,t)$ uses an edge that is not in E, which has been excluded from all triangulations with dilation less than ρ; a contradiction. E' can be computed by repeatedly computing shortest s-t-paths π; as long as $\pi < \rho d(s,t)$, we find an edge $e \notin T$ on π, add e to E' and forbid it in future paths. The number of edges bounds the number of iterations of this process; using the comparison oracle, we can efficiently perform each iteration. ◀

2.3 Incremental algorithm

Based on the SAT formulation and the algorithm for the dilation path separation problem, IncMDT is simple. Given an initial triangulation T with dilation ρ, we enumerate the set of candidate edges and construct a triangulation supergraph G with bound ρ. We construct the initial SAT formula M and solve it; if it is unsatisfiable, the initial triangulation is optimal. Otherwise, we repeat the following until the model becomes unsatisfiable or we encounter an edge conflict, keeping track of the best triangulation found, see Figure 2.

We extract the new triangulation T' from the SAT solver and compute the dilation ρ' and a pair s, t of points realizing ρ'. If ρ' is better than the best previously found dilation ρ, we update ρ and mark all edges e with $\vartheta(e) \geq \rho'$ as *impossible*. We then set $T = T'$ and solve the dilation path separation problem for ρ, G, T, s and t. We add the resulting clause to M and let the SAT solver find a new solution.

2.4 Binary search

Preliminary experiments with IncMDT showed that we spend almost all runtime for computing dilations, even for instances for which we could rely exclusively on interval arithmetic, requiring no exact computations. For many instances, most iterations of IncMDT resulted in tiny improvements of the dilation. To reduce the number of iterations (and thus, dilations computed), we considered the binary search-based algorithm BinMDT.

At any point in time, aside from the dilation ρ_{ub} of the best known triangulation, BinMDT maintains a lower bound ρ_{lb} on the dilation, initialized as described in the full version. As long as $\rho_{\mathrm{ub}} - \rho_{\mathrm{lb}} \geq \sigma$ for a small threshold value σ, BinMDT performs a binary search. It computes a new dilation bound $\rho = \frac{1}{2}(\rho_{\mathrm{lb}} + \rho_{\mathrm{ub}})$. It then uses the SAT model in a similar way as IncMDT to determine whether a triangulation T with $\rho(T) < \rho$ exists. If it does, it updates $\rho_{\mathrm{ub}} = \rho(T)$; otherwise, it updates $\rho_{\mathrm{lb}} = \rho$. Once $\rho_{\mathrm{ub}} - \rho_{\mathrm{lb}}$ falls below σ, BinMDT falls back to a slightly modified version of IncMDT to find the MDT, starting from the best known triangulation with dilation ρ_{ub}. For more details, see also the full version.

3 Empirical evaluation

Now we present experiments to evaluate our algorithms. Code and data are publicly available[1]. We used Python 3.12, with a core module written in C++20 for all computationally heavy tasks; the code was compiled with GCC 13.2.0 in release mode. We use CGAL 5.6.1 for geometric primitives and exact number types, Boost 1.83 for utility functions and pybind11 2.12 for Python bindings and use the incremental SAT solver CaDiCaL 1.9.5 via the PySAT interface for solving the SAT models. All experiments were performed on Linux workstations equipped with AMD Ryzen 9 7900 CPUs with 12 cores/24 threads and 96 GiB of DDR5-5600 RAM running Ubuntu 24.04.1.

Experiment design We collected and generated a large set of instances, consisting of instances from the following instance classes. In all cases, the coordinates of points in the instances are either integers or double precision floating-point numbers.

random-small [3] The 210 instances (30 for each size $n \in \{10, 20, \ldots, 70\}$) were generated by placing uniformly random points inside a 10×10 square.

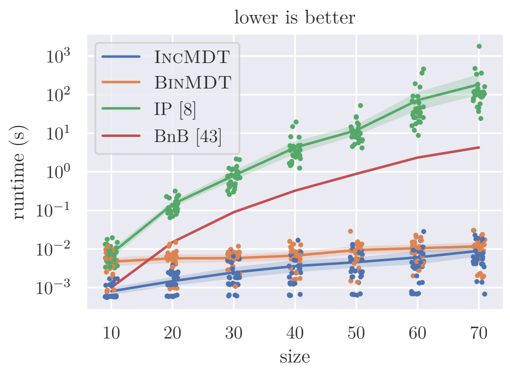
random Two sets of randomly generated instances (total of 800 instances) with points with float coordinates chosen uniformly between 0 and 10^3, ranging from 50 to 10,000 points.

public [7, 6, 20, 19, 10] Well-known publicly available point sets used in the CG:SHOP challenges [7, 6], TSPLIB instances [19], instances from a VLSI dataset [20] and point sets from the Salzburg Database of Polygonal Inputs [10]. In total, we collected 486 instances with up to 10,000 points and an additional 38 with up to 30,000 points.

Comparison to state of the art We compare our approaches to two exact algorithms for the MDT. Note that both use floating-point arithmetic and are not guaranteed to find the optimal solution (although we can confirm that all previous solutions are within a small relative error). The first approach is the IP approach from [3] and the second is the branch & bound (BnB) algorithm from [21].

For *random-small*, both IncMDT and BinMDT outperform the IP and BnB approach by a large margin (up to four orders of magnitude), see Figure 3. All instances are solvable

[1] Code and data: `https://doi.org/10.5281/zenodo.14266122`



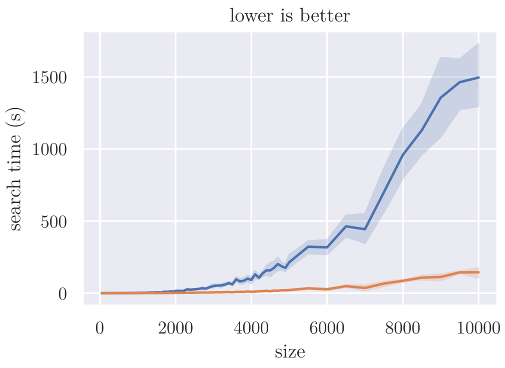
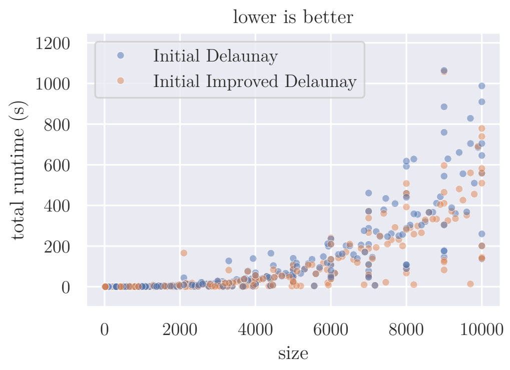


Figure 3 (Left) Runtime comparison with the approaches from [3] and [21] on the *random-small* set. **(Right)** BinMDT is significantly faster than IncMDT on the *random* benchmark set.



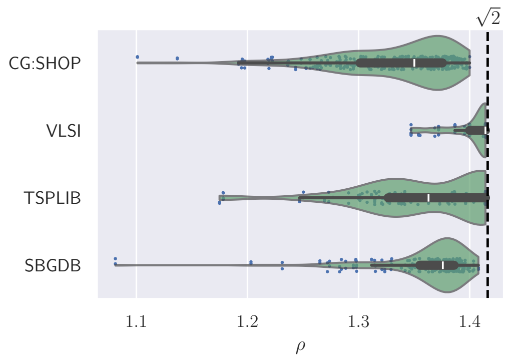


Figure 4 Experiments on the *public* benchmark set. **(Left)** Using the improved Delaunay triangulation as an initial solution significantly improved the performance. **(Right)** The dilation of the MDTs is at most $\sqrt{2}$ for all instances.

in less than 0.1 s. Additionally, [21] provided results for TSPLIB [19] instances (part of our *public* instance set) with up to 200 points. Our approach is significantly faster than theirs, solving each of these instances to provable optimality in less than 1 s instead of up to 1248 s; a table with all instances and runtimes can be found in the full version.

Algorithm comparison We now compare IncMDT to BinMDT; see the full version for more detail. We conduct our first experiment on the *random* instances with up to 10,000 points; this experiment confirms that BinMDT achieves a significantly lower runtime. For more details, see Figure 3.

We also conduct an additional experiment on the *public* instances up to 10,000 points to determine whether performing greedy, local improvements to the Delaunay triangulation, which we use as initial solution, is worthwhile; see Figure 4. The improved Delaunay triangulation significantly reduces the runtime of BinMDT for almost all instances. Detailed results for all *public* instances, as well as an additional experiment on the *public* instance set showing that BinMDT can solve instances with up to 30,000 points in less than 17 h to provable optimality, can be found in the full version.

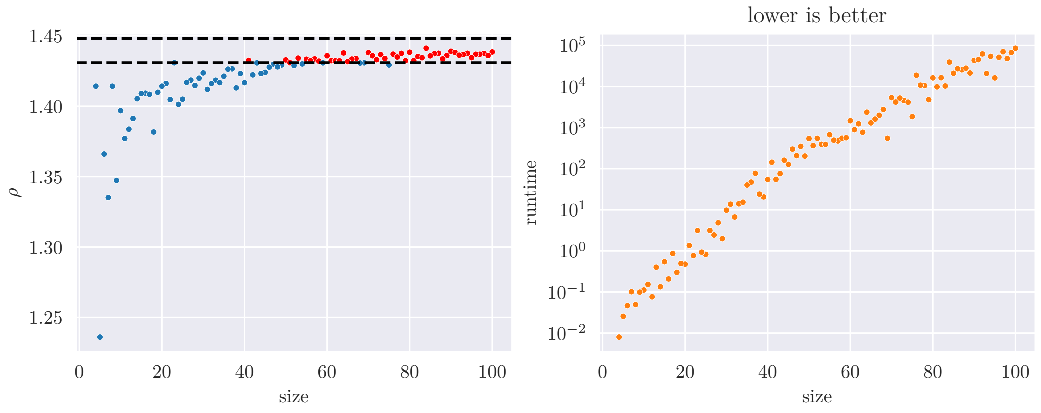


Figure 5 Dilations and runtimes for regular n-gons for $4 \leq n \leq 100$. Red dots improve the current lower bound of 1.4308 that comes from the regular 23-gon. The dashed black lines mark the known upper bound of 1.4482 and the previous best lower bound of 1.4308.

Regular n-gons The worst-case dilation of a regular n-gon has received considerable attention [17, 9, 22], with a lower bound of 1.4308 and an upper bound of 1.4482. Improving this gap is an open question posed by [9], originating from [2, 13]. With our exact algorithm, we were able to compute bounds for $n \in \{4, 5, \ldots, 100\}$ and found that the dilation of a regular 84-gon is at least 1.44116, see Figures 1 and 5 and the full version for details.

4 Conclusion

We have presented exact algorithms for minimum dilation triangulations, greatly outperforming previous methods from the literature. This has also yielded insights into the intricate structure of optimal solutions for regular n-gons, together with new lower bounds on the worst-case dilation of triangulations. This demonstrates the value of computational tools for gaining analytic insights that seem out of reach with purely manual analysis.

References

1 Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. In *Computing in Euclidean geometry*, pages 47–123. World Scientific, 1995.

2 Prosenjit Bose and Michiel H. M. Smid. On plane geometric spanners: A survey and open problems. *Comput. Geom.*, 46(7):818–830, 2013. URL: `https://doi.org/10.1016/j.comgeo.2013.04.002`, `doi:10.1016/J.COMGEO.2013.04.002`.

3 Aléx F. Brandt, Miguel M. Gaiowski, Cid C. de Souza, and Pedro J. de Rezende. Minimum dilation triangulation: Reaching optimality efficiently. In *Proceedings of the 26th Canadian Conference on Computational Geometry, CCCG 2014, Halifax, Nova Scotia, Canada, 2014*. Carleton University, Ottawa, Canada, 2014. URL: `http://www.cccg.ca/proceedings/2014/papers/paper09.pdf`.

4 Otfried Cheong, Herman J. Haverkort, and Mira Lee. Computing a minimum-dilation spanning tree is NP-hard. *Comput. Geom.*, 41(3):188–205, 2008. `doi:10.1016/J.COMGEO.2007.12.001`.

5 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications, 3rd Edition.* Springer, 2008. URL: `https://www.worldcat.org/oclc/227584184`.

6 Erik D Demaine, Sándor P Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Computing convex partitions for point sets in the plane: The CG:SHOP Challenge 2020. *arXiv preprint arXiv:2004.04207*, 2020.

7 Erik D Demaine, Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Area-optimal simple polygonalizations: The CG Challenge 2019. *Journal of Experimental Algorithmics (JEA)*, 27(2):1–12, 2022.

8 Maria Gisela Dorzán, Mario Guillermo Leguizamón, Efrén Mezura-Montes, and Gregorio Hernández-Peñalver. Approximated algorithms for the minimum dilation triangulation problem. *J. Heuristics*, 20(2):189–209, 2014. doi:10.1007/S10732-014-9237-2.

9 Adrian Dumitrescu and Anirban Ghosh. Lower bounds on the dilation of plane spanners. *Int. J. Comput. Geom. Appl.*, 26(2):89–110, 2016. doi:10.1142/S0218195916500059.

10 Günther Eder, Martin Held, Steinþór Jasonarson, Philipp Mayer, and Peter Palfrader. Salzburg database of polygonal data: Polygons and their generators. *Data in Brief*, 31:105984, 2020. doi:10.1016/j.dib.2020.105984.

11 David Eppstein. Spanning trees and spanners. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 425–461. North Holland / Elsevier, 2000. doi:10.1016/B978-044482537-7/50010-3.

12 Panos Giannopoulos, Rolf Klein, Christian Knauer, Martin Kutz, and Dániel Marx. Computing geometric minimum-dilation graphs is NP-hard. *Int. J. Comput. Geom. Appl.*, 20(2):147–173, 2010. doi:10.1142/S0218195910003244.

13 Iyad Kanj. Geometric spanners: Recent results and open directions. In *Third International Conference on Communications and Information Technology, ICCIT 2013*, pages 78–82. IEEE, 2013. doi:10.1109/ICCITECHNOLOGY.2013.6579526.

14 Alexander Klein. Effiziente Berechnung einer dilationsminimalen Triangulierung, 2006.

15 Christian Knauer and Wolfgang Mulzer. An exclusion region for minimum dilation triangulations. In *(Informal) Proceedings of the 21st European Workshop on Computational Geometry (EuroCG 2005)*, pages 33–36. Technische Universiteit Eindhoven, 2005. URL: http://www.win.tue.nl/EWCG2005/Proceedings/9.pdf.

16 László Kozma. Minimum average distance triangulations. In Leah Epstein and Paolo Ferragina, editors, *20th Annual European Symposium on Algorithms (ESA 2012)*, volume 7501 of *Lecture Notes in Computer Science*, pages 695–706. Springer, 2012. doi:10.1007/978-3-642-33090-2_60.

17 Wolfgang Mulzer. Minimum dilation triangulations for the regular n-gon. *Master's Thesis. Freie Universität Berlin, Germany.*, 2004.

18 Giri Narasimhan and Michiel H. M. Smid. *Geometric spanner networks.* Cambridge University Press, 2007.

19 G. Reinelt. TSPLIB–A Traveling Salesman Problem Library. *ORSA Journal of Computing*, 3(4):376–384, 1991.

20 A. Rohe. VLSI data set, 2013. URL: https://www.math.uwaterloo.ca/tsp/vlsi/index.html.

21 Sattar Sattari and Mohammad Izadi. An exact algorithm for the minimum dilation triangulation problem. *J. Glob. Optim.*, 69(2):343–367, 2017. doi:10.1007/S10898-017-0517-X.

22 Sattar Sattari and Mohammad Izadi. An improved upper bound on dilation of regular polygons. *Comput. Geom.*, 80:53–68, 2019. doi:10.1016/J.COMGEO.2019.01.009.

23 Israel Vite Silva, Nareli Cruz Cortés, Gregorio Toscano Pulido, and Luis Gerardo de la Fraga. Optimal triangulation in 3D computer vision using a multi-objective evolutionary algorithm. In *Applications of Evolutinary Computing, EvoWorkshops 2007*, volume 4448 of *Lecture Notes in Computer Science*, pages 330–339. Springer, 2007. doi:10.1007/978-3-540-71805-5_36.

24 Victor J. D. Tsai. Delaunay triangulations in TIN creation: An overview and a linear-time algorithm. *Int. J. Geogr. Inf. Sci.*, 7(6):501–524, 1993. doi:10.1080/02693799308901979.

25 Chun-Hsien Wu, Kuo-Chuan Lee, and Yeh-Ching Chung. A delaunay triangulation based method for wireless sensor network deployment. *Comput. Commun.*, 30(14-15):2744–2752, 2007. doi:10.1016/J.COMCOM.2007.05.017.

26 Hongyu Zhou, Hongyi Wu, Su Xia, Miao Jin, and Ning Ding. A distributed triangulation algorithm for wireless sensor networks on 2d and 3d surface. In *30th IEEE International Conference on Computer Communications (INFOCOM 2011)*, pages 1053–1061. IEEE, 2011. doi:10.1109/INFCOM.2011.5934879.

A Branch-and-Bound Algorithm for the Traveling Salesman Problem with Neighborhoods

Sándor P. Fekete[1], Rouven Kniep[1], Dominik Krupke[1], and Michael Perk[1]

1 **Department of Computer Science, TU Braunschweig**
s.fekete@tu-bs.de, {kniep, krupke, perk}@ibr.cs.tu-bs.de

Abstract

The Traveling Salesman Problem with Neighborhoods (TSPN) generalizes the classical Traveling Salesman Problem by requiring a tour to visit polygonal regions rather than fixed points, a natural goal that arises in various applications. While the geometric TSP allows arbitrarily close approximation and provably optimal solutions for benchmark instances of significant size, the TSPN is considerably more challenging, both in theory (due to APX-hardness) and practice, for which only benchmark instances up to 16 regions have been solved to provable optimality. In this paper, we propose a branch-and-bound algorithm that solves polygonal TSPN instances to optimality. Through computational experiments on 500 benchmark instances with 50 polygons each, our method achieves a 85.6 % optimality rate within 60 s. We also explore the impact of key design choices, providing insights into effective solution strategies for TSPN.

1 Introduction

A natural generalization of the classical Traveling Salesman Problem (TSP) is the Traveling Salesman Problem with Neighborhoods (TSPN), which asks for a shortest roundtrip that visits each of a given family $P_1, \ldots, P_n$ of regions in the plane, see Figure 1 for an example.

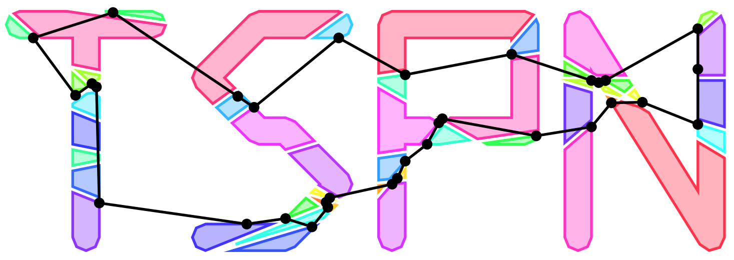


Figure 1 Example TSPN instance with a feasible solution.

While the geometric TSP allows both polynomial-time approximation schemes [22, 4] and solution to provable optimality for point sets of considerable size (such as the 85 900-city instance solved by [1, 8]), the TSPN is considerably more challenging, both in theory (with APX-hardness [10, 11]), and practice (with the state of the art being provably optimal solutions for benchmark instances up to 16 regions [17]).

Contribution. We present a branch-and-bound algorithm that solves TSPN instances to provable optimality. Across 500 benchmark instances with 50 polygons each, 85.6 % are solved to optimality within 60 s, significantly advancing the state of the art [17]. We also evaluate the impact of various algorithmic design choices.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.

Related Work. A special case of TSPN is the Close-Enough TSP (CE-TSP), where each neighborhood is a circle, which is also related to the Lawn Mowing Problem [2, 3, 14, 15]. Branch-and-bound algorithms have been studied for CE-TSP [12, 9]; however, the TSPN allows arbitrary (including non-convex) neighborhoods, making it more general. Many heuristics and approximation algorithms for the TSPN rely on assumptions (e.g., fatness [23], disjoint neighborhoods [10, 11], or comparable region diameters [13]) to manage its APX-hardness. Specialized approaches address specific settings such as aerial vehicle routing [20], and hybrid methods combine meta-heuristics with TSP solvers [25]. Non-convex Mixed Integer Nonlinear Programming (MINLP) formulations have been proposed for TSPN, including symmetric and asymmetric variants [17], with algorithmic improvements to reduce solution times. However, computational tests were limited to smaller or convex neighborhoods. Other work derives approximations and bounds for the metric TSPN using the Minimum Spanning Tree with Neighborhoods [7].

2 Branch-and-Bound Algorithm

Our algorithm (Algorithm 1) builds on the branch-and-bound framework by Coutinho et al. [9] for the CE-TSP, improving and extending it to handle the TSPN. We begin by constructing a root node using a universal ordering on a subset of the polygons (see Section 2.2). From there, we branch by selecting a polygon that is not yet visited in the current solution and creating a new branch for each possible insertion position (Section 2.3).

For the sequence of polygons in each node, we solve a Second-Order Cone Program (SOCP) to obtain the optimal tour for that ordering (Section 2.1). If this tour intersects all polygons, it is a feasible solution (see Figure 2); otherwise, the SOCP value provides a valid lower bound to prune the node if a superior solution is already known. Because evaluating a single node can yield multiple child nodes, we apply a search strategy to decide which node to process next (Section 2.4).

Throughout the search, the algorithm keeps track of the incumbent (best known feasible) solution and the node with the lowest lower bound, which together define the current optimality gap[1]. Search is terminated if this gap is smaller than a desired threshold, or if all nodes are explored.

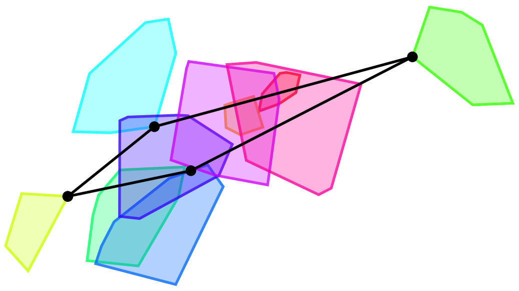


Figure 2 A sequence with only 4 polygons is feasible for this example instance with 10 polygons.

[1] Measured as the relative difference to the lower bound, $^{ub}/_{lb} - 1$.

Algorithm 1 Branch-and-Bound Algorithm

```
Require: Set of polygons $\mathcal{I}$
 1: Preprocess and simplify $\mathcal{I}$.
 2: $\mathcal{Q} \leftarrow [\,]$                                  ▷ Queue of leaf nodes to explore
 3: $\mathcal{Q}$.push(GETROOTNODE($\mathcal{I}$))                    ▷ See Section 2.2
 4: $ub \leftarrow \infty$
 5: while $\mathcal{Q} \neq \emptyset \land \min\{v'.lb \mid v' \in \mathcal{Q}\} < ub$ do
 6:     $v \leftarrow$ GETNEXTNODE($\mathcal{Q}$)              ▷ Search strategy, see Section 2.4
 7:     Remove $v$ from $\mathcal{Q}$
 8:     if $v.lb \geq ub$ then
 9:         continue
10:     end if
11:     if Solution in $v$ covers all polygons in $\mathcal{I}$ then
12:         $ub \leftarrow v.lb$
13:     else
14:         for $child \in$ BRANCH($v$) do                     ▷ Branching, see Section 2.3
15:             $\mathcal{Q}$.push($child$)
16:         end for
17:     end if
18: end while
19: return Incumbent solution corresponding to $ub$, or $\perp$ if no solution was found.
```

2.1 Touring a Sequence of Polygons

▶ **Theorem 2.1.** *Let $P_0, \ldots, P_{n-1} \subset \mathbb{R}^2$ be convex polygons. Then the shortest tour visiting these polygons in order can be computed in polynomial time.*

Proof. A convex polygon P_i can be represented by linear constraints $S_i(x, y)$. We formulate the problem as a Second-Order Cone Program (SOCP), which can be solved in polynomial time via interior-point methods [5]. For each i, introduce a point $(x_i, y_i) \in \mathbb{R}^2$ constrained by $S_i(x_i, y_i)$, ensuring $(x_i, y_i) \in P_i$. For readability, all indices are assumed modulo n.

To encode the total tour length, let $d_i \geq 0$ be the distance between (x_i, y_i) and (x_{i+1}, y_{i+1}). We introduce auxiliary variables $\hat{x}_i, \hat{y}_i$ subject to

$$\hat{x}_i \geq x_i - x_{i+1}, \quad \hat{x}_i \geq x_{i+1} - x_i, \quad \hat{y}_i \geq y_i - y_{i+1}, \quad \hat{y}_i \geq y_{i+1} - y_i,$$

and impose second-order cone constraints $d_i^2 \geq \hat{x}_i^2 + \hat{y}_i^2$. Minimizing $\sum_{i=0}^{n-1} d_i$ can be done in polynomial time, and yields the shortest tour as d_i will be tight in the optimal solution. ◀

To handle non-convex polygons P'_i, including those with holes, we propose an alternative constraint set $S'_i(x, y)$ that employs binary variables and can be expressed as a Mixed Integer Second-Order Cone Program (MISOCP). Suppose each P'_i can be decomposed into a finite set of convex polygons $\mathcal{R}_i$, see Figure 3. We introduce a binary variable $r_c \in \mathbb{B}$ for each $c \in \mathcal{R}_i$ and require exactly one of these polygons to be active by enforcing $\sum_{c \in \mathcal{R}_i} r_c = 1$. For each convex polygon c, let $S_i^c(x, y)$ denote its associated constraints. We then ensure $(x, y) \in P'_i$ by imposing the implications $r_c \implies S_i^c(x, y)$ for all $c \in \mathcal{R}_i$. These implications can be enforced via indicator constraints (available in many solvers) or through Big-M linearizations, where M can be limited by the bounding box of P'_i.

For polygons without holes, a minimum convex decomposition can be computed in polynomial time [18, 6]. For polygons with holes, a decomposition is always feasible by triangu-

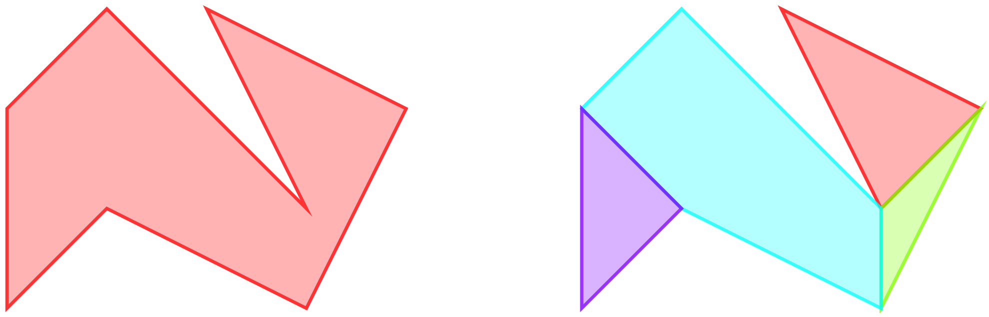


Figure 3 A non-convex polygon (left) and its decomposition in convex areas (right).

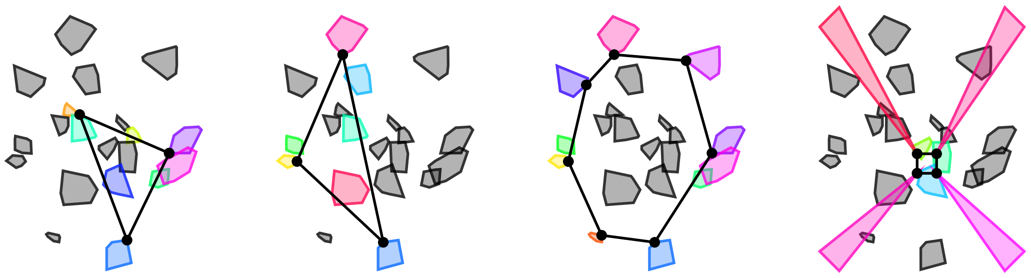


Figure 4 Root node strategies (left to right): Random, longest edge + farthest polygon, convex hull, and an example demonstrating poor convex hull performance.

lation, but finding a minimal decomposition is NP-hard [21]. Solving the resulting MISOCP is also NP-hard, and the use of indicator variables or Big-M constraints can lead to weak relaxations, making these methods computationally expensive in practice. Thus, we will investigate later to lazily decompose the polygons in our branch-and-bound algorithm, instead of using the MISOCP formulation directly.

2.2 Root Node

The root node's initial polygon sequence must be extendable to an optimal solution. Any sequence of up to three polygons trivially satisfies this condition. We evaluate three methods for selecting these polygons: **Random:** Select any three polygons at random. **Longest Edge+Farthest Polygon (LEFP):** Pick two polygons with the largest pairwise distance, then add the polygon farthest from these two. **Convex Hull (CHR):** Exploit the observation that any set of disjoint polygons on the instance's convex hull must appear in the same order in some optimal solution. Although this strategy may include more than three polygons in the root sequence, it can perform poorly on certain instances. For example, if the polygons are strongly protruding inwards, they may not force the initial tour to span the region effectively. Figure 4 illustrates all three strategies, including a case where CHR performs poorly.

2.3 Branching

When the (partial) tour in a node does not cover all polygons, we branch by selecting a missing polygon and creating a new branch for each possible insertion position. Figure 5 illustrates this approach. We compare two strategies for choosing the polygon to branch on: **Random**: Select a missing polygon uniformly at random. **Farthest Polygon:** Select the polygon that lies farthest from the current tour.

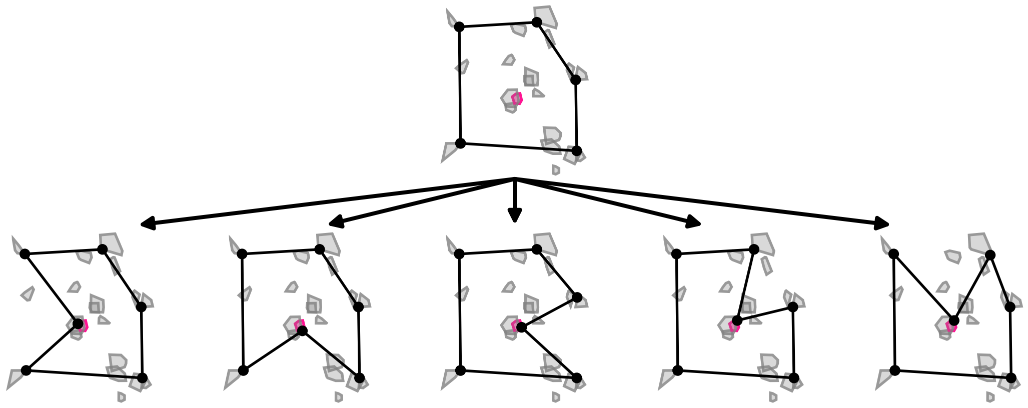


Figure 5 Branching on the insertion position of the selected farthest polygon.

Because each polygon is initially replaced by its convex hull for efficiency, a polygon may appear in the partial sequence but still be effectively unvisited. If the MISOCP formulation is used, we can simply replace the convex hull with the polygon itself. Otherwise, we must branch on this polygon by decomposing it into convex parts and creating one branch per part (see Figure 6). This ensures that every leaf node includes the polygon, with at least one leaf containing its optimal hitting point.

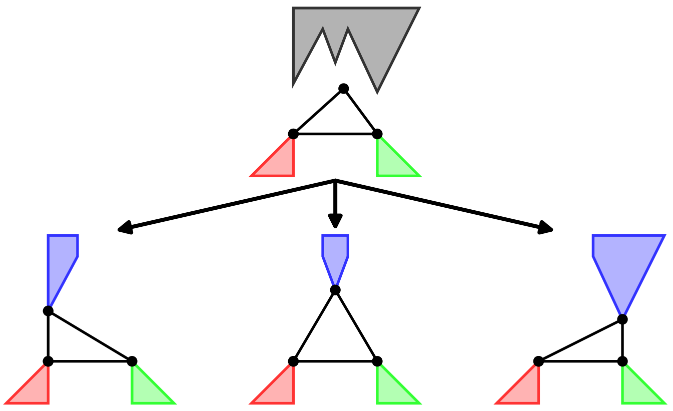


Figure 6 Branching on a non-convex polygon by decomposing it into convex pieces.

2.4 Search Strategies

We implement four different strategies to select the next node for exploration during the branch-and-bound search: **Random:** Pick a node uniformly at random from the queue.

BFS: Choose the node with the best, i.e., smallest, lower bound. **DFS:** Continue exploring the best child of the current node, aiming to quickly achieve feasibility and find a better incumbent solution. **DFS+BFS:** Initially proceed like DFS, but whenever a node is pruned or a new feasible solution is discovered, sort the queue by lower bounds. This combines the fast incumbent improvements of DFS with the tighter lower-bound focus of BFS.

3 Experiments

In the following, we investigate how various algorithmic choices affect performance. We tested 500 instances, each containing 50 random polygons: 45 % convex, 45 % concave (up to 10 units), and 10 % larger concave polygons with holes (up to 20 units), see Figure 7 for examples. The instance `n50_ps70_001` is used for the convergence plots throughout this section. We regard a solution as optimal if its optimality gap is below 0.01 % within 60 s.

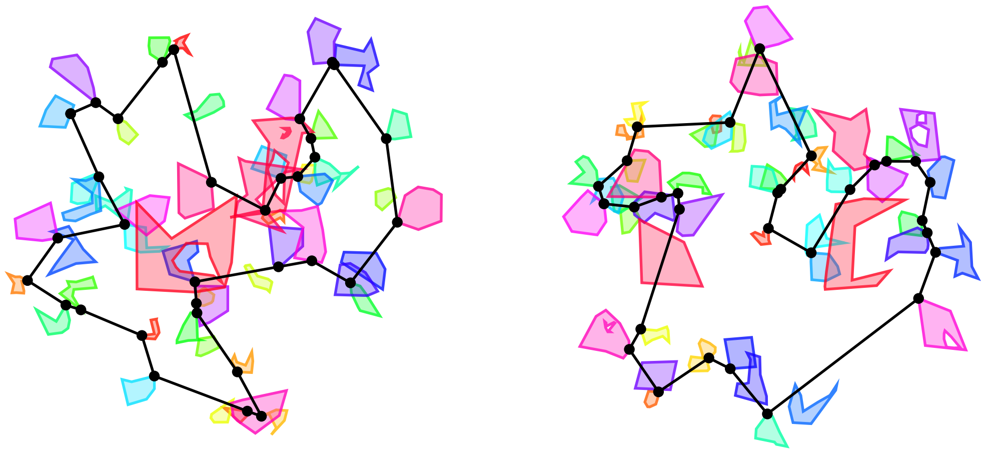


Figure 7 Optimal solutions for `n50_ps70_001` (left) and `n50_ps70_002` (right).

Our algorithm is implemented in C++ and uses Gurobi 12.0 [19] to solve the mathematical programs. Geometry operations rely on Boost.Geometry 1.83 [16] and CGAL 6.0.1 [24] with exact predicates and constructions. We compiled using `g++` 13.3 and ran all tests on an AMD Ryzen 7900 workstation with 96 GiB of DDR5-5200 RAM under Ubuntu 24.04.

Preprocessing Initial preprocessing to simplify the instances showed a modest improvement on some instances and improved average runtimes slightly. However, no additional instances were solved in time; see Figure 8.

Warm Start Heuristically computing an initial solution only benefited the random search strategy (Figure 9). For other strategies, the high upfront cost (usually between 10 s to 60 s) of the naive algorithm used outweighed gains.

Root Node The choice of root node sequence critically affects performance (Figure 10). Using a convex hull root node (CHR) was generally fastest, while a LEFP approach performed better on instances with large polygons. A random strategy is not recommended.

Search Strategy DFS found solutions fastest, whereas BFS increases the lower bound fastest. A combined DFS+BFS strategy is a good compromise; see Figure 11. It excels when slightly relaxed optimality tolerances are acceptable.

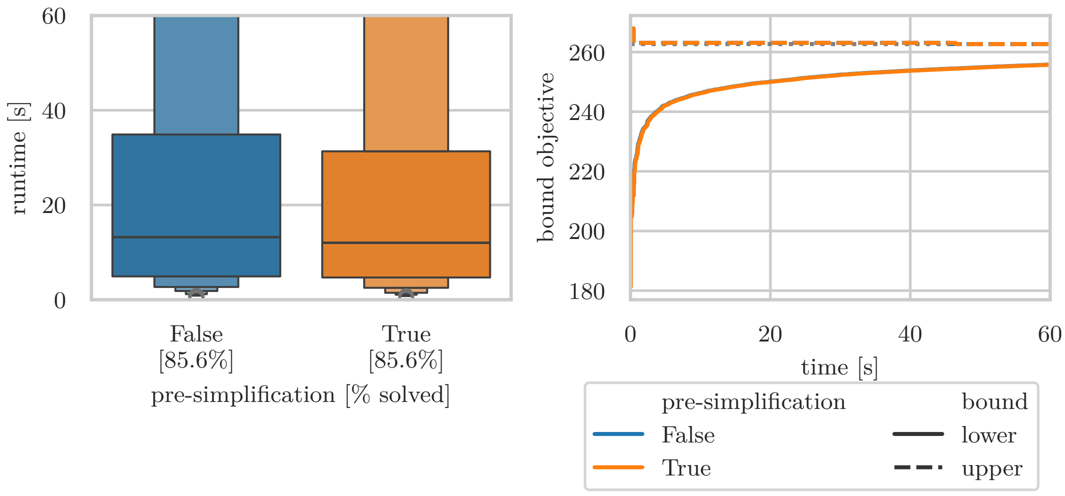


Figure 8 Runtime (left) and bound convergence (right) for different preprocessing settings. Sometimes pre-simplification improved initial bounds and sped up convergence.

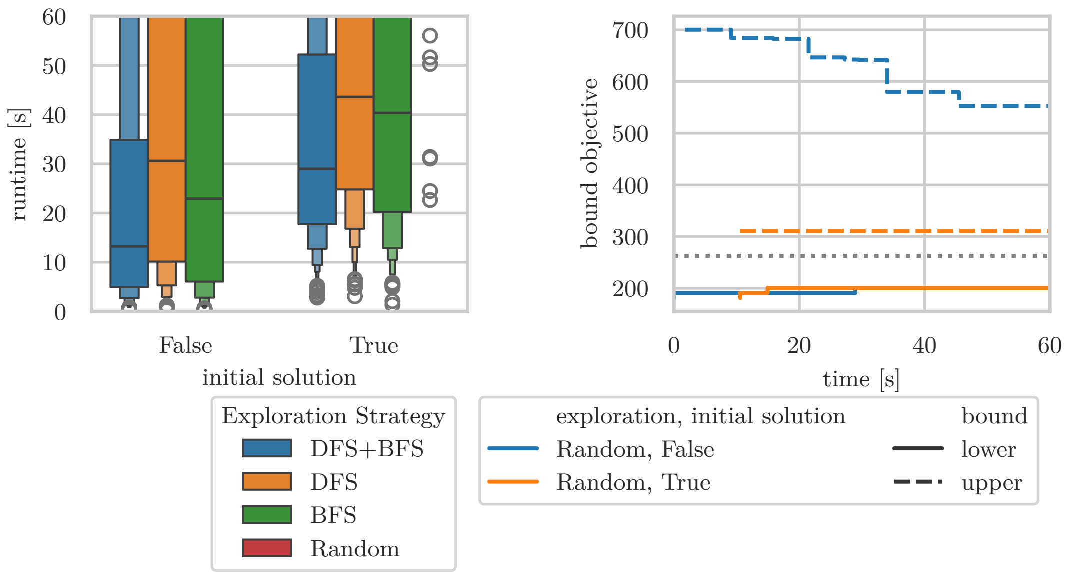


Figure 9 Runtime analysis (left) and random node exploration bound convergence (right) for different initial solutions. Other exploration strategies showed no improvement from warm starts.

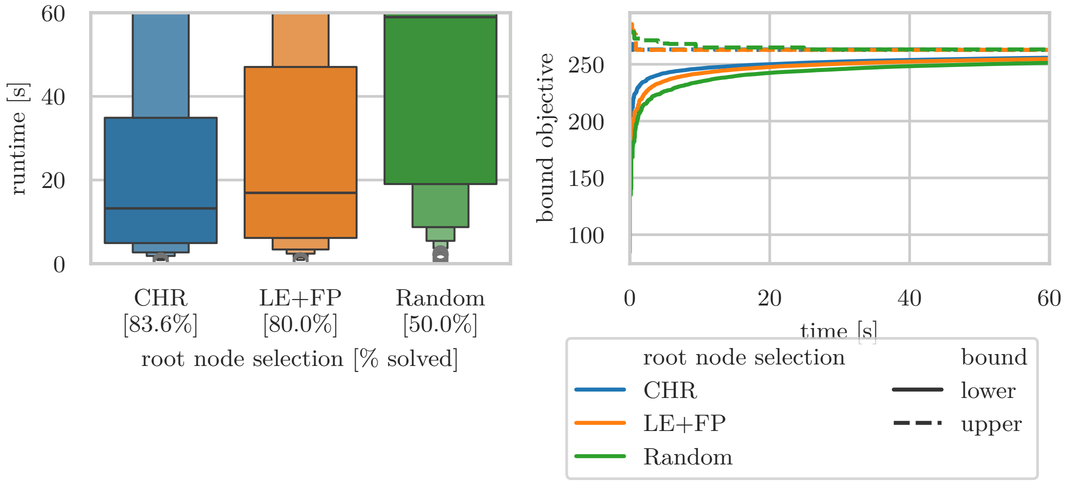


Figure 10 Runtime (left) and bound convergence (right) for different root node choices. CHR performed best overall; however, for larger polygons, LEFP was more robust.

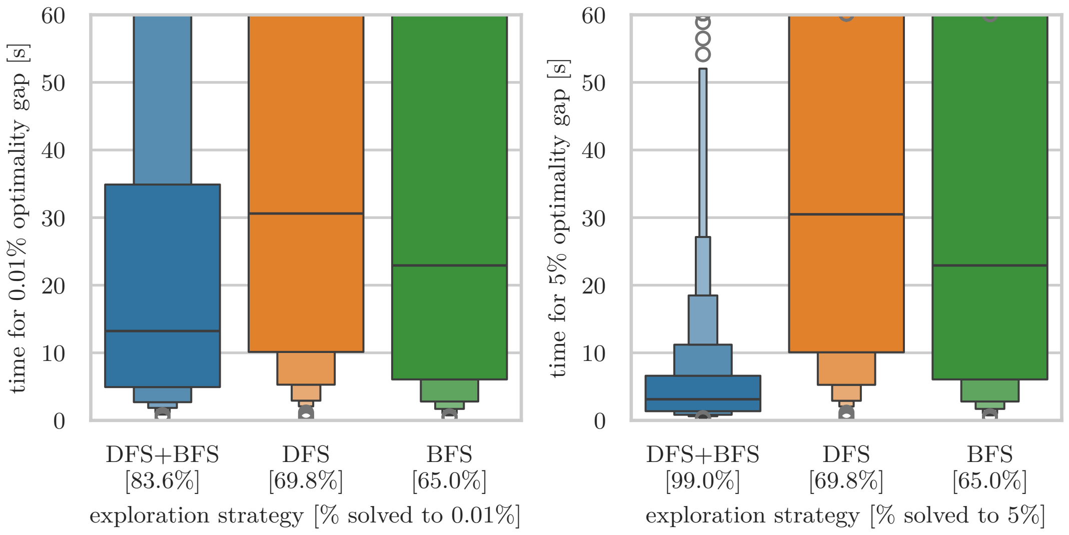


Figure 11 Runtime for 0.01 % (left) and 5 % (right) optimality tolerance using different search strategies. When allowing a 5 % gap, DFS+BFS terminates quickly.

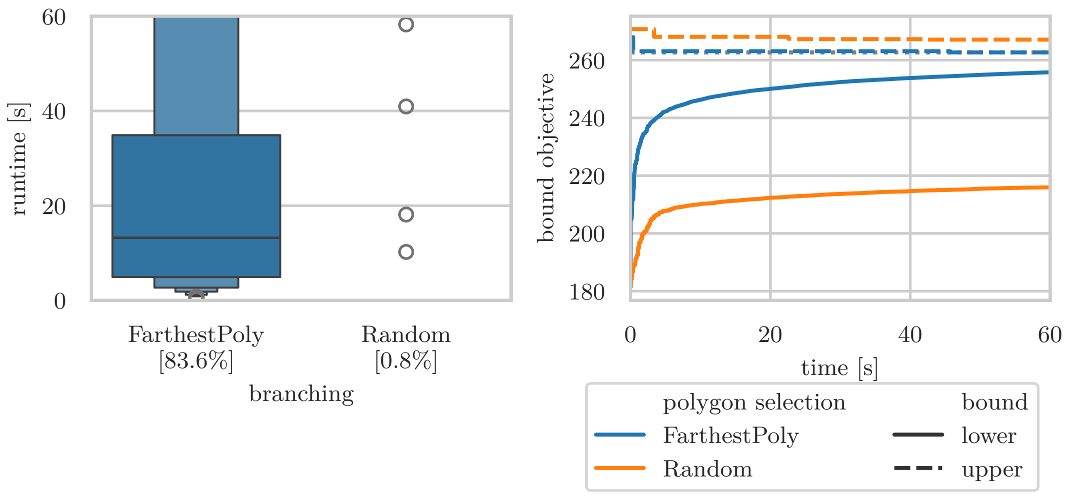


Figure 12 Runtime (left) and bound convergence (right) for different polygon selection methods. Farthest polygon significantly improved performance over random.

Polygon Selection When selecting the next polygon to branch on, choosing the *farthest polygon* consistently outperformed random selection (Figure 12).

Decomposition Modeling When handling non-convex polygons, decomposition branching yields faster and more reliable performance than indicator modeling in Gurobi (Figure 13). Although indicator modeling is simpler to implement, it leads to weaker relaxations.

Optimality Tolerance Relaxing the optimality gap significantly reduces runtime for gaps of 5 % to 10 %; see Figure 14. Differences between 0.01 % and 0.1 % are negligible, but a gap of 5 % or higher often saves substantial time.

Threats to Validity All solutions were validated, and a set of unit tests ensured correctness of core components. Results were checked for consistency between upper and lower bounds. Moreover, the selected 500 instances may not be fully representative of real-world scenarios, though the set provides diversity by fixing instance size and varying polygon shapes.

4 Conclusion and Future Work

In this paper, we presented a branch-and-bound algorithm for the TSPN and evaluated the impact of various design decisions on its performance. While most of the results aligned with our expectations, it was surprising to find that manually branching on the decomposition of non-convex polygons outperformed handling them with Gurobi. Looking ahead, we have several ideas for further improvements, including enhanced early-pruning strategies and more advanced parallelization techniques.

Acknowledgments

The implementation builds on code from an enhanced branch-and-bound algorithm for the CE-TSP developed by Dominik Krupke, Barak Ugav, and Michael Perk.

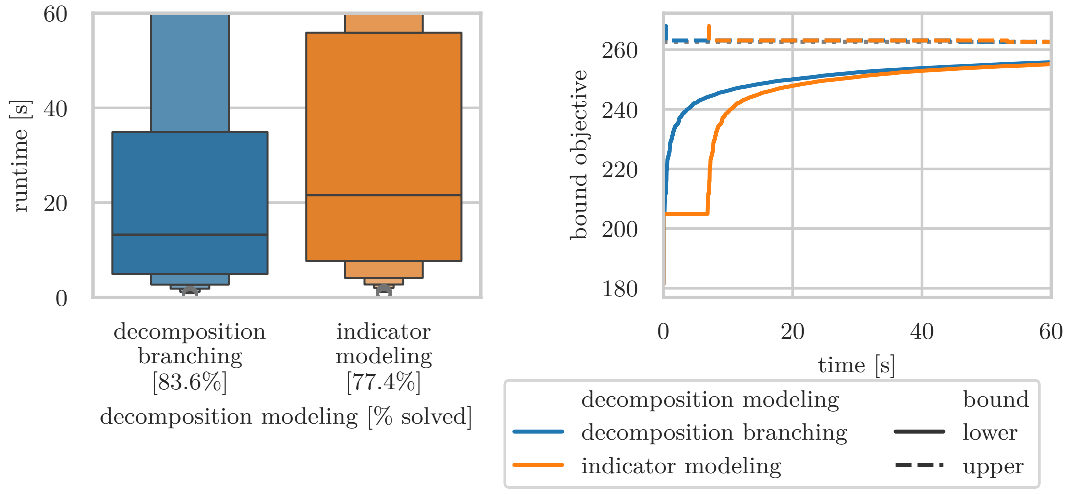


Figure 13 Runtime (left) and bound convergence (right) for different modeling approaches. Decomposition branching converged faster, whereas indicator modeling slowed early convergence.

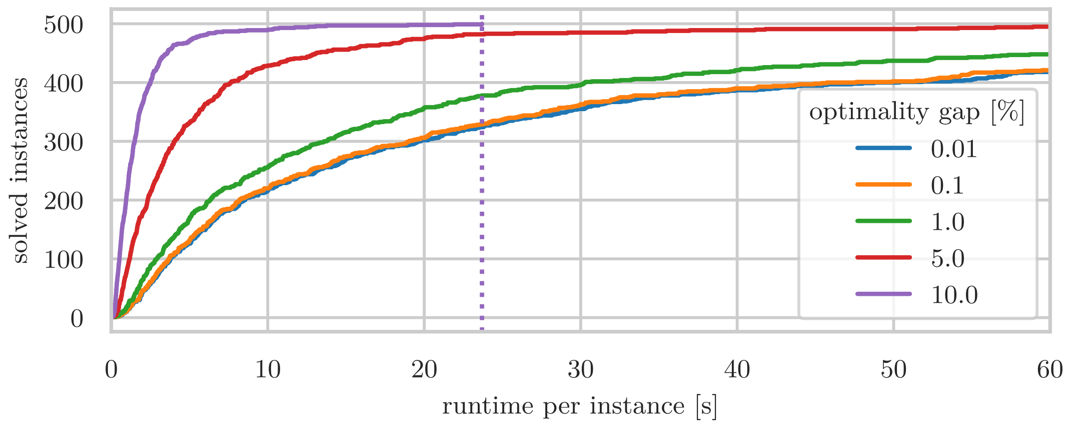


Figure 14 Number of instances solved within a given gap versus time. Wider gaps of 5 % to 10 % substantially reduce computation time.

References

1 David L. Applegate, Robert E. Bixby, Vasek Chvátal, William J. Cook, Daniel G. Espinoza, Marcos Goycoolea, and Keld Helsgaun. Certification of an optimal TSP tour through 85, 900 cities. *Oper. Res. Lett.*, 37(1):11–15, 2009. doi:10.1016/J.ORL.2008.09.006.

2 Esther M. Arkin, Sándor P. Fekete, and Joseph S. B. Mitchell. The lawnmower problem. In *Canadian Conference on Computational Geometry (CCCG)*, pages 461–466, 1993. URL: https://cglab.ca/~cccg/proceedings/1993/Paper79.pdf.

3 Esther M. Arkin, Sándor P. Fekete, and Joseph S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17:25–50, 2000. doi:10.1016/S0925-7721(00)00015-8.

4 Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998. doi:10.1145/290179.290180.

5 S.P. Boyd and L. Vandenberghe. *Convex Optimization*. Number pt. 1 in Berichte über verteilte messysteme. Cambridge University Press, 2004. URL: https://books.google.de/books?id=mYm0bLd3fcoC.

6 Bernard Chazelle and David P. Dobkin. Optimal convex decompositions. In Godfried T. Toussaint, editor, *Computational Geometry*, volume 2 of *Machine Intelligence and Pattern Recognition*, pages 63–133. North-Holland, 1985. doi:10.1016/B978-0-444-87806-9.50009-8.

7 Andrew Clark. A submodular optimization approach to the metric traveling salesman problem with neighborhoods. In *58th IEEE Conference on Decision and Control, CDC*, pages 3383–3390. IEEE, 2019. doi:10.1109/CDC40024.2019.9030031.

8 William J Cook, David L Applegate, Robert E Bixby, and Vasek Chvatal. *The traveling salesman problem: a computational study*. Princeton University Press, 2011.

9 Walton Pereira Coutinho, Roberto Quirino do Nascimento, Artur Alves Pessoa, and Anand Subramanian. A branch-and-bound algorithm for the close-enough traveling salesman problem. *INFORMS J. Comput.*, 28(4):752–765, 2016. doi:10.1287/IJOC.2016.0711.

10 Mark de Berg, Joachim Gudmundsson, Matthew J. Katz, Christos Levcopoulos, Mark H. Overmars, and A. Frank van der Stappen. TSP with neighborhoods of varying size. In Rolf H. Möhring and Rajeev Raman, editors, *10th Annual European Symposium ESA*, volume 2461, pages 187–199. Springer, 2002. doi:10.1007/3-540-45749-6_20.

11 Mark de Berg, Joachim Gudmundsson, Matthew J. Katz, Christos Levcopoulos, Mark H. Overmars, and A. Frank van der Stappen. TSP with neighborhoods of varying size. *J. Algorithms*, 57(1):22–36, 2005. doi:10.1016/J.JALGOR.2005.01.010.

12 Andrea Di Placido, Claudia Archetti, Carmine Cerrone, and Bruce Golden. The generalized close enough traveling salesman problem. *European Journal of Operational Research*, 310(3):974–991, 2023. doi:10.1016/j.ejor.2023.04.010.

13 Adrian Dumitrescu and Joseph SB Mitchell. Approximation algorithms for tsp with neighborhoods in the plane. *Journal of Algorithms*, 48(1):135–159, 2003.

14 Sándor P. Fekete, Dominik Krupke, Michael Perk, Christian Rieck, and Christian Scheffer. A closer cut: Computing near-optimal tours for the lawn mowing problem. In *Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 1–14, 2023. doi:10.1137/1.9781611977561.

15 Sándor P. Fekete, Dominik Krupke, Michael Perk, Christian Rieck, and Christian Scheffer. The lawn mowing problem: From algebra to algorithms. In *European Symposium on Algorithms (ESA)*, pages 45:1–45:18, 2023. doi:10.4230/LIPIcs.ESA.2023.45.

16 Barend Gehrels, Bruno Lalande, Mateusz Loskot, Adam Wulkiewicz, Menelaos Karavelas, and Fisikopoulos. Vissarion. Boost geometry 1.82, 2023. URL: https://www.boost.org/.

17 Iacopo Gentilini, François Margot, and Kenji Shimada. The travelling salesman problem with neighbourhoods: MINLP solution. *Optimization Methods Software*, 28(2):364–378, 2013. doi:10.1080/10556788.2011.648932.

18 Daniel H Greene. The decomposition of polygons into convex parts, manuscript. *Computational geometry*, 1:235–259, 1983.

19 Gurobi Optimization LLC. Gurobi Optimizer 10.0. https://www.gurobi.com/, 2023.

20 Dae-Sung Jang, Hyeok-Joo Chae, and Han-Lim Choi. Optimal control-based UAV path planning with dynamically-constrained TSP with neighborhoods. *CoRR*, 2016. doi:10.48550/arXiv.1612.06008.

21 Andrzej Lingas. The power of non-rectilinear holes. In Mogens Nielsen and Erik Meineche Schmidt, editors, *Automata, Languages and Programming, 9th Colloquium*, volume 140, pages 369–383. Springer, 1982. doi:10.1007/BFB0012784.

22 Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999. doi:10.1137/S0097539796309764.

23 Joseph S. B. Mitchell. A PTAS for TSP with neighborhoods among fat regions in the plane. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 11–18. Society for Industrial and Applied Mathematics, 2007.

24 The CGAL Project. CGAL 5.6. https://www.cgal.org, 2023.

25 Bo Yuan and Tiantian Zhang. Towards solving TSPN with arbitrary neighborhoods: A hybrid solution. In Markus Wagner, Xiaodong Li, and Tim Hendtlass, editors, *Artificial Life and Computational Intelligence - Third Australasian Conference, ACALCI*, volume 10142, pages 204–215, 2017. doi:10.1007/978-3-319-51691-2_18.

Getting Better All the Time? Harmonic and Cumulative Traveling Salesman Problems

Sándor P. Fekete[1], Dominik Krupke[1], Christian Rieck[2], Arne Schmidt[1], and Tobias Wallner[1]

1 **Department of Computer Science, TU Braunschweig, Braunschweig, Germany**
`s.fekete@tu-bs.de, {krupke, aschmidt, wallner}@ibr.cs.tu-bs.de`

2 **Department of Discrete Mathematics, University of Kassel, Kassel, Germany**
`christian.rieck@mathematik.uni-kassel.de`

Abstract

We consider versions of the Traveling Salesman Problem in which the cost of travel changes along the way: In the Harmonic TSP (HTSP), the cost of each traversed edge is its length divided by the number of previously visited vertices, while in the Cumulative TSP (CTSP), it is its length multiplied by this number. Both problems are related to the Minimum Latency Problem, which aims at minimizing the average arrival time of a tour; as we show (along with geometric properties), these problems are distinct. In addition to observations placing these variants into context, our main contribution is a constant-factor approximation for the HTSP on general metric instances.

1 Introduction

The Traveling Salesman Problem (TSP) is a classical problem of combinatorial optimization, seeking a cheapest round trip that visits each of a given set of points. In its original formulation, the cost of a TSP tour is simply the sum of edge lengths; however, in many applications from transportation and logistics, the price of traversing an edge also depends on the context within a tour: In ride-sharing or group travel, the cost per passenger may decrease as additional riders join the route and share expenses. Conversely, in some pick-up and freight services, the total load grows over the course of the route, increasing fuel consumption and handling costs. This leads to variants in which the travel cost changes along the way, either *inversely proportionally*, referred to as the Harmonic variant (HTSP), or *proportionally* (the Cumulative variant (CTSP)) to the number of previously visited vertices.

Numerous variants with travel cost that change dynamically have been studied before, including the Discounted-Reward TSP [9], the Time-dependent TSP, and the Time-dependent Vehicle Routing Problem [1, 2, 3, 11, 16, 21]. Most closely related to the variants of this paper is the well-studied Minimum Latency Problem (MLP) of minimizing the average arrival time of a Hamiltonian path that visits a given set of points. Blum et al. [8] gave a 144-approximation algorithm for metric instances. Goemans and Kleinberg [15] improved the factor to 10.78, Archer and Williamson [4] to 9.28, and Chaudhuri et al. [10] to 3.59 with run-time $\widetilde{\mathcal{O}}(n^4)$. Arora and Karakostas [6] also presented a quasi-polynomial time approximation scheme for the special case of traveling on a tree.

As part of our contribution, we observe that the MLP, the HTSP and the CTSP are distinct. Furthermore, optimal solutions for the Cumulative TSP and the Minimum Latency Problem are always within constant factors of each other, whereas this is generally not true for the Harmonic TSP and the MLP.

Formally, we define the Harmonic and Cumulative TSP variants as follows.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.

Harmonic Traveling Salesman Problem (HTSP)

Given: A connected, weighted graph $G = (V, E)$ with n vertices, a starting vertex $v_0 \in V$, and a weight function $c\colon E \to \mathbb{R}_0^+$.
Wanted: A permutation $\pi\colon \{0, ..., n-1\} \to V$, with $\pi(0) = v_0$, such that the total cost

$$C = \sum_{i=1}^{n-1} \left[\frac{\ell(\pi(i-1), \pi(i))}{i} \right] + \frac{\ell(\pi(n-1), \pi(0))}{n}$$

is minimized, with $\ell(v_i, v_j)$ denoting the weight of a shortest path from v_i to v_j.

Cumulative Traveling Salesman Problem (CTSP)

Given: A connected, weighted graph $G = (V, E)$ with n vertices, a starting vertex $v_0 \in V$, and a weight function $c\colon E \to \mathbb{R}_0^+$.
Wanted: A permutation $\pi\colon \{0, ..., n-1\} \to V$, with $\pi(0) = v_0$, such that the total cost

$$C = \sum_{i=1}^{n-1} [i \cdot \ell(\pi(i-1), \pi(i))] + n \cdot \ell(\pi(n-1), \pi(0))$$

is minimized, with $\ell(v_i, v_j)$ denoting the weight of a shortest path from v_i to v_j.

In both variants, any solution corresponds to a sequence of edges obtained by concatenating the shortest paths from $\pi(i)$ to $\pi(i+1)$ for $i = 0, ..., n-1$. When traversing an edge e of weight $c(e)$ as part of a shortest path from $\pi(k)$ to $\pi(k+1)$, we say that the vertices $\pi(0), ..., \pi(k)$ have already been *collected* at this time. Traversing e increases the cost of the current tour by an amount of $1/k \cdot c(e)$ or $k \cdot c(e)$, respectively. We refer to $1/k$ or k as the current *speed factor*, which depends on the number k of previously collected vertices.

Our Contributions We provide the following insights and results.

1. We discuss the relationship to the Minimum Latency Problem, as well as geometric properties of optimal tours, and show that both CTSP and HTSP can be solved by a simple dynamic programming approach on trees with a fixed number of leaves.
2. We establish a constant-factor approximation algorithm for the HTSP.

Throughout the remainder of this paper, we assume that the triangle inequality holds.

2 Basic Observations

Relationship to the Minimum Latency Problem Like the HTSP and the CTSP, the MLP asks for a permutation π of the vertices minimizing a cost function, the *latency* $L = \sum_{i=1}^{n-1}(n-i) \cdot \ell(v_{i-1}, v_i)$; closing the tour by returning to v_0 does not incur any further latency. Thus, any solution to the MLP can also be considered as a solution to the HTSP or the CTSP. We now evaluate the respective quality of the solutions.

The instance depicted in Figure 1 shows that no constant or logarithmic approximation factor can be guaranteed for the HTSP by using an optimal MLP solution: For both problems, only two options are reasonable: (a) first visiting all the vertices on the left, followed by those on the right, or (b) vice versa. For the HTSP, the total cost of (a) is in $\Theta(\ln k)$, while that of (b) is in $\Theta(k)$, making (a) optimal for large k. However, (b) is optimal for the MLP for any k: in particular, the total latency of (a) is $7k^2/2 + 9k/2$ whereas that of (b) is $7k^2/2 + 3k/2$.

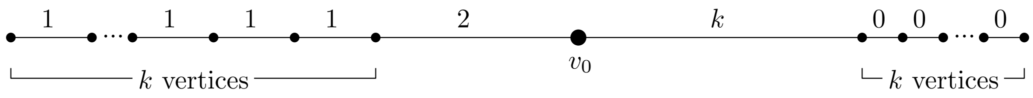


Figure 1 An instance showing that the MLP provides no approximation for the HTSP.

An optimal MLP solution does not yield an optimal CTSP solution either, even though the CTSP cost of a permutation and the MLP latency of the reversed permutation only differ by the simple tour length: Consider the permutation $(v_0, ..., v_{n-1})$ and its inverse as solutions for the CTSP and the MLP, respectively. The cost of the CTSP solution is $C = \sum_{i=1}^{n-1} (i \cdot \ell(v_{i-1}, v_i)) + n \cdot \ell(v_{n-1}, v_0)$. The total latency of the MLP solution is $L = (n-1) \cdot \ell(v_0, v_{n-1}) + \sum_{j=1}^{n-1}(n-j-1) \cdot \ell(v_{n-j}, v_{n-j-1}) = \sum_{i=1}^{n-1}(i-1) \cdot \ell(v_{i-1}, v_i) + (n-1) \cdot \ell(v_{n-1}, v_0) = C - \left(\sum_{i=1}^{n-1} \ell(v_{i-1}, v_i) + \ell(v_{n-1}, v_0)\right)$.

Figure 2 illustrates an instance for which the optimal CTSP and the inverted MLP solutions are distinct: Out of the 24 possible permutations, v_0, v_4, v_2, v_3, v_1 with $C = 75$ is optimal for the CTSP; the optimal MLP solution is v_0, v_1, v_2, v_3, v_4 with $L = 47$.

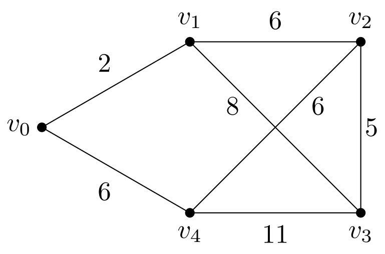


Figure 2 A simple instance showing that the MLP and the CTSP are not equivalent.

However, the inverse of an optimal solution for the MLP yields a 3-approximation for the CTSP. Based on the MLP solution, from $L = C - \left(\sum_{i=1}^{n-1} \ell(v_{i-1}, v_i) + \ell(v_{n-1}, v_0)\right)$, $\sum_{i=1}^{n-1} \ell(v_{i-1}, v_i) \leq L$, $L \leq C$, and $\ell(v_{n-1}, v_0) \leq L$, we may infer $L \leq C \leq 3L$. Assume that $(v_0, ..., v_{n-1})$ is an optimal solution for the MLP with total latency L_{OPT}. Inverting this permutation yields a CTSP solution with cost $C \leq 3 \cdot L_{\text{OPT}} \leq 3 \cdot C_{\text{OPT}}$.

Dynamic Program for Trees with Bounded Number of Leaves Paths as in Figure 1 and trees with a bounded number of leaves can be solved in polynomial time via dynamic programming. For the HTSP, consider a path $v_l, ..., v_0 (= w_0), ..., w_r$ with l vertices on the left and r on the right of the root $v_0 = w_0$. Define $\ell_m(u, u')$ as the cost of moving from u to u' after m vertices have been visited (with no additional speed factor changes), with the exact form of ℓ_m depending on the problem variant.

For $0 \leq i \leq l$ and $0 \leq j \leq r$, let $s_{\underline{i},j}$ be the minimum accumulated cost of a partial tour that has visited the first i vertices on the left and the first j vertices on the right of the root, ending at (and thus newly visiting) v_i. Define $s_{i,\underline{j}}$ analogously for tours ending at w_j. To compute $s_{\underline{i},j}$, we consider the two possible predecessors v_{i-1} and w_j of v_i:

$$s_{\underline{i},j} = \begin{cases} 0 & \text{if } i = 0 \text{ and } j = 0, \\ \bot, & \text{if } i < 0 \text{ or } j < 0, \\ \min\left\{s_{\underline{i-1},j} + \ell_{i+j-1}(v_{i-1}, v_i),\ s_{i-1,\underline{j}} + \ell_{i+j-1}(w_j, v_i)\right\}, & \text{otherwise.} \end{cases}$$

We can compute $s_{i,\underline{j}}$ analogously, yielding $\mathcal{O}(n^2)$ states in total, each requiring $\mathcal{O}(1)$ time to compute. Having visited all $l+r$ non-root vertices, we return to $v_0 = w_0$, so the total cost is

$$\min\Big\{ s_{\underline{l},r} + \ell_{l+r}(v_l, v_0), \quad s_{l,\underline{r}} + \ell_{l+r}(w_r, w_0) \Big\}.$$

A standard backtracking procedure (storing which minimum was chosen) recovers the tour in linear time. Analogous arguments are applicable to the CTSP.

More generally, for a tree with k leaves, one extends the state to k dimensions (one per leaf), each corresponding to a branch of the tree. The first i vertices collected by an optimal HTSP solution will form an interval (including the root vertex) in each of the branches. In the CTSP, after the first i vertices have been collected by an optimal solution, the vertices not yet collected will form an interval in each branch. These properties can be used to formulate dynamic programs, considering the cost of visiting an additional vertex in any of the branches, and leading to a run-time of $\mathcal{O}(k \cdot n^k)$.

Edge Crossings for Geometric Instances of the HTSP Due to the triangle inequality, optimal solutions for the Euclidean TSP never contain any crossing edges, reducing possible solutions to the set of simple polygonizations, whose worst-case cardinality is between $\Omega(4.642^n)$ [12] and $\mathcal{O}(54.543^n)$ [19, 20], and thus considerably less than the full set of $(n-1)!$ permutations. Simple connectedness is also a prerequisite for several geometric approximation techniques, such as the PTASs by Mitchell [17] and Arora [5].

However, simplicity does not hold for geometric instances of the HTSP; in fact, there may be $\Theta(n^2)$ edge crossings in an optimal solution. We demonstrate this by constructing a set of n points in a polar coordinate system as follows. We define the polar radius r_i of v_i as $r_i := 2^{i^2}$, forcing the optimal solution to visit the vertices in the order $v_0, ..., v_{n-1}, v_0$, regardless of their respective polar angles. It is straightforward to show that visiting any vertex v_j with $j > i$ before v_i results in a suboptimal total cost, as the additional cost incurred by collecting v_i first is overcompensated by the resulting speed factor improvement for the exponentially larger distance to v_j. Now that the sequence of the vertices v_i in the optimal solution has been determined based on their polar radius, we construct their polar angles φ_i, ensuring that the solution contains $\Theta(n^2)$ crossings. For $i = 1, 2, 3, 4$, we set φ_i to $\frac{3\pi}{2}$, 0, $\frac{\pi}{2}$, and π. For $i > 4$, we alternatingly set φ_i to $0 + \varepsilon_i$ and $\pi + \varepsilon_i$ with increasing $\varepsilon_i > 0$, such that every edge $v_{i-1}v_i$, each chosen in the optimal solution, crosses every edge $v_0v_1, v_1v_2, ..., v_{i-3}v_{i-2}$. An example of this construction is shown in Figure 3.

3 Constant-Factor Approximation Algorithm for the HTSP

We propose an approximation technique for the HTSP by applying existing approximation algorithms for the k-TSP and the k-MST problem; this has some resemblance to the approach used by Blum et al. [8] for the MLP, but yields a different analysis and approximation factor.

Let $\pi = (v_0, v_1, ..., v_n)$ be a permutation of V with optimal total weight, and let ℓ_i abbreviate the weight $\ell(v_{i-1}, v_i)$ of a shortest path p_i from v_{i-1} to v_i. Consider an alternative solution π' formed by altering π in the following way: After reaching the 2^j-th (new) vertex of π, $j = 0, 1, ..., \log n$, we retrace our path back to v_0. We then repeat the process, following the same sequence of vertices, until we reach the 2^{j+1}-th vertex of π. The arrows in Figure 4 illustrate this construction.

▶ **Lemma 3.1.** *The total cost of a solution π' constructed from an optimal solution π as described above is at most 6 times as high as the cost of π.*

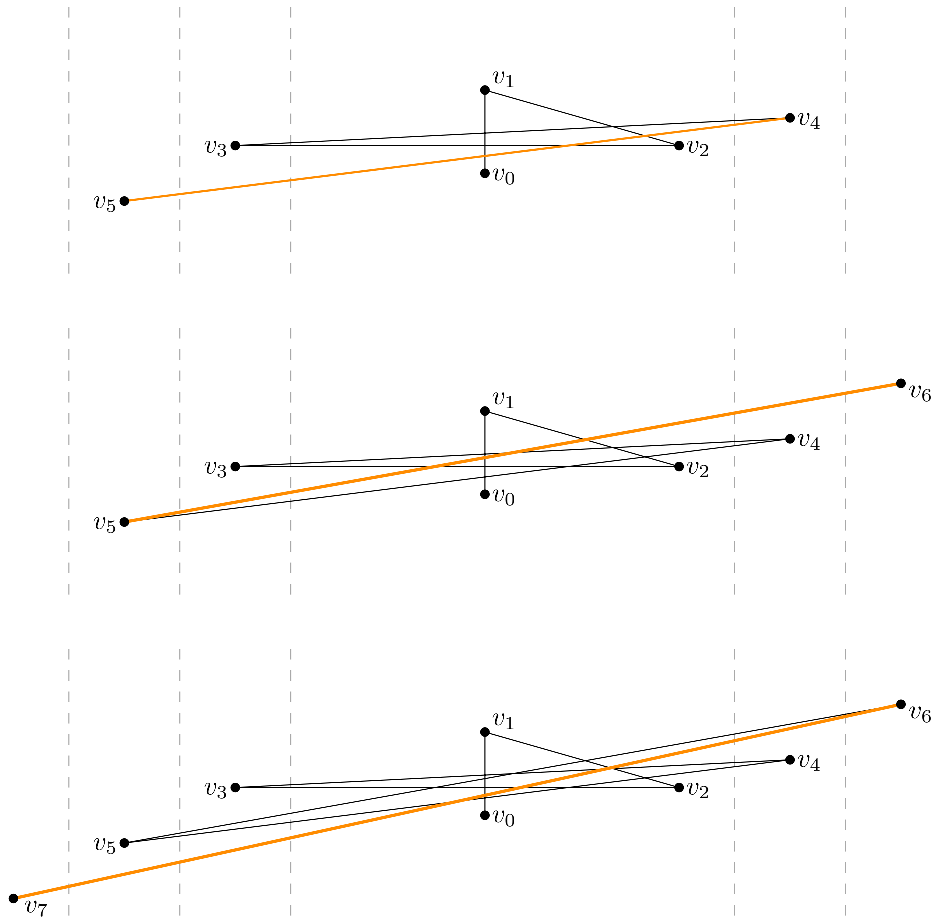


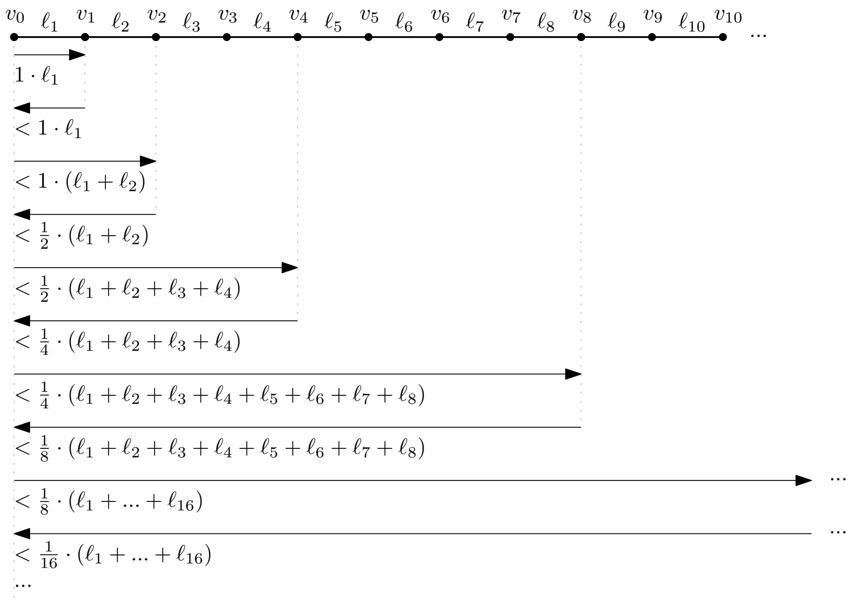
Figure 3 Instance (not to scale) with $\Theta(n^2)$ crossings in an optimal solution. The latest chosen edge is highlighted in orange, and the dashed lines represent large distances, growing like 2^{i^2}.

Proof. For each path p_i with length ℓ_i, we compare the sum of the speed factors with which p_i is traversed in π' to the speed factor of p_i in an optimal solution.

Given the decreasing speed factor, we may bound the total cost $C_{\pi'}$ of π'. Summing up the speed factors of the p_i (as illustrated in Figure 4) yields:

$$\begin{aligned}C_{\pi'} \leq\ & \ell_1\Big(3\cdot 1 + 2\cdot\frac{1}{2} + 2\cdot\frac{1}{4} + 2\cdot\frac{1}{8} + \ldots + 2\cdot\frac{1}{2^{\lceil\log n\rceil}}\Big)\\ &+ \ell_2\Big(1 + 2\cdot\frac{1}{2} + 2\cdot\frac{1}{4} + 2\cdot\frac{1}{8} + \ldots + 2\cdot\frac{1}{2^{\lceil\log n\rceil}}\Big)\\ &+ \ell_3\Big(1\cdot\frac{1}{2} + 2\cdot\frac{1}{4} + 2\cdot\frac{1}{8} + \ldots + 2\cdot\frac{1}{2^{\lceil\log n\rceil}}\Big)\\ &+ \ell_4\Big(1\cdot\frac{1}{2} + 2\cdot\frac{1}{4} + 2\cdot\frac{1}{8} + \ldots + 2\cdot\frac{1}{2^{\lceil\log n\rceil}}\Big)\\ &+ \ell_5\Big(1\cdot\frac{1}{4} + 2\cdot\frac{1}{8} + \ldots + 2\cdot\frac{1}{2^{\lceil\log n\rceil}}\Big)\\ &+ \ldots\end{aligned}$$

Rewriting the geometric series yields



v_0 ℓ_1 v_1 ℓ_2 v_2 ℓ_3 v_3 ℓ_4 v_4 ℓ_5 v_5 ℓ_6 v_6 ℓ_7 v_7 ℓ_8 v_8 ℓ_9 v_9 ℓ_{10} v_{10} ...
$1 \cdot \ell_1$
$< 1 \cdot \ell_1$
$< 1 \cdot (\ell_1 + \ell_2)$
$< \frac{1}{2} \cdot (\ell_1 + \ell_2)$
$< \frac{1}{2} \cdot (\ell_1 + \ell_2 + \ell_3 + \ell_4)$
$< \frac{1}{4} \cdot (\ell_1 + \ell_2 + \ell_3 + \ell_4)$
$< \frac{1}{4} \cdot (\ell_1 + \ell_2 + \ell_3 + \ell_4 + \ell_5 + \ell_6 + \ell_7 + \ell_8)$
$< \frac{1}{8} \cdot (\ell_1 + \ell_2 + \ell_3 + \ell_4 + \ell_5 + \ell_6 + \ell_7 + \ell_8)$
$< \frac{1}{8} \cdot (\ell_1 + ... + \ell_{16})$
$< \frac{1}{16} \cdot (\ell_1 + ... + \ell_{16})$
...

Figure 4 Illustration for the cost analysis of π'.

$$\begin{aligned} C_{\pi'} \leq \ell_1 + \ell_1 \cdot \left(2 \sum_{k=0}^{\lceil \log n \rceil} \left(\frac{1}{2}\right)^k\right) + \ell_2 \cdot \left(-1 + 2 \sum_{k=0}^{\lceil \log n \rceil} \left(\frac{1}{2}\right)^k\right) \\ + \ell_3 \cdot \left(-\frac{1}{2} + 2 \sum_{k=0}^{\lceil \log n \rceil} \left(\frac{1}{2}\right)^k - 2 \sum_{k=0}^{0} \left(\frac{1}{2}\right)^k\right) + ... + \\ + \ell_j \cdot \left(-\frac{1}{2^{\lceil \log j \rceil - 1}} + 2 \sum_{k=0}^{\lceil \log n \rceil} \left(\frac{1}{2}\right)^k - 2 \sum_{k=0}^{\lceil \log j \rceil - 2} \left(\frac{1}{2}\right)^k\right) + ... \end{aligned}$$

With the convergence of the geometric series, the coefficient of ℓ_j in this equation becomes

$$\begin{aligned} & -\frac{1}{2^{\lceil \log j \rceil - 1}} + 2 \sum_{k=0}^{\lceil \log n \rceil} \left(\frac{1}{2}\right)^k - 2 \sum_{k=0}^{\lceil \log j \rceil - 2} \left(\frac{1}{2}\right)^k \\ & \leq -\left(\frac{1}{2}\right)^{\lceil \log j \rceil - 1} + 2 \cdot 2 - 2 \cdot \left(2 - \left(\frac{1}{2}\right)^{\lceil \log j \rceil - 2}\right) \\ & \leq -2 \left(\frac{1}{2}\right)^{\lceil \log j \rceil} + 8 \left(\frac{1}{2}\right)^{\lceil \log j \rceil} \leq 6 \left(\frac{1}{2}\right)^{\lceil \log j \rceil} \leq 6 \cdot \frac{1}{j}. \end{aligned}$$

Thus, each path weight ℓ_j is weighted at most 6 times as much as in π (in which it is weighted with $1/j$). It is straightforward to establish this bound for the coefficients of ℓ_1 and ℓ_2. ◀

For each round k, we need to visit 2^{k-1} vertices and return to v_0 while bounding the cost with respect to the k-th round of π'. Let C_k be its total cost (accounting for speed factors) and λ_k its simple weight (only totaling edge weights). We analyze two options for finding a round-trip from v_0 via k additional vertices.

The first variant uses the related k-MST problem, asking for a tree of minimum total length spanning exactly k vertices. In the *rooted* variant, we are given a vertex v_0 that has to be included in the spanning tree. Clearly, the (unrooted) k-MST problem can be reduced to the rooted k-MST problem by going through all of the n possible roots. The problem has been proven to be NP-complete [7, 18]. Garg [14] provides a polynomial-time 2-approximation for this problem in metric graphs.

▶ **Theorem 3.2.** *There is a 24α-approximation for the HTSP in metric graphs, with α denoting an approximation factor for the rooted k-MST problem.*

Proof. Clearly, the total weight $\lambda_{\text{MST},k}$ of a v_0-rooted $(2^{k-1}+1)$-MST is not higher than λ_k. Each rooted $(2^{k-1}+1)$-MST yields a tour by traveling every edge exactly twice. Consider the solution resulting from traversing, one after the other, the v_0-rooted $(2^{k-1}+1)$-MST for $k = 1, 2, \ldots$. Then, when traversing the $(2^{k-1}+1)$-MST, at least $2^{k-2}+1$ (1 in case of the 2-MST, i.e., $k = 1$) have been collected before. Hence, the cost $C_{\text{MST},k}$ of traversing this particular tree is $C_{\text{MST},k} \leq \frac{1}{2^{k-2}+1} \cdot 2\lambda_{\text{MST},k}$. In the k-th round of π', the best speed factor is at least $\frac{1}{2^{k-1}+1}$. So, the cost $C_{\pi',k}$ of the round is $C_{\pi',k} \geq \frac{1}{2^{k-1}+1}\lambda_k \geq \frac{1}{2^{k-1}+1}\lambda_{\text{MST},k}$. Thus, we obtain $C_{\text{MST},k} \leq \frac{1}{2^{k-2}+1} \cdot 2\lambda_{\text{MST},k} \leq 2 \cdot \frac{2^{k-1}+1}{2^{k-2}+1} \cdot C_{\pi',k} \leq 4 \cdot C_{\pi',k}$. So, we can 4-approximate π'. With that, Lemma 3.1 yields the theorem. ◀

▶ **Corollary 3.3.** *There is a 48-approximation for the HTSP in metric graphs.*

The arguments for using a k-MST algorithm to approximate a round of π' can be adapted to using an algorithm for k-TSP, asking for a minimum cost tour visiting exactly k vertices; clearly the problem is NP-hard [7]. Garg [13] gave a 3-approximation for the rooted variant of the k-TSP in metric graphs.

By applying the same arguments as in the k-MST approach, but without the factor 2 for traversing the spanning trees, we obtain the following.

▶ **Theorem 3.4.** *There is a 12α-approximation for the HTSP in metric graphs, with α denoting an approximation factor for the rooted k-TSP problem.*

Proof. Because the v_0-rooted $(2^{k-1}+1)$-TSP solution is the shortest tour visiting $2^{k-1}+1$ vertices including v_0, its total weight $\lambda_{\text{TSP},k}$ is not higher than λ_k. We concatenate the v_0-rooted $(2^{k-1}+1)$-TSP tours, with cost $C_{\text{TSP},k}$ each, for $k = 1, 2, \ldots$.

By $C_{\text{TSP},k} \leq \frac{1}{2^{k-2}+1} \cdot \lambda_{\text{TSP},k}$ and $C_{\pi',k} \geq \frac{1}{2^{k-1}+1} \cdot \lambda_k \geq \frac{1}{2^{k-1}+1} \cdot \lambda_{\text{TSP},k}$, we obtain $C_{\text{TSP},k} \leq \frac{1}{2^{k-2}+1} \cdot \lambda_{\text{TSP},k} \leq \frac{2^{k-1}+1}{2^{k-2}+1} \cdot C_{\pi',k} \leq 2 \cdot C_{\pi',k}$. By combining this with Lemma 3.1, the theorem follows. ◀

▶ **Corollary 3.5.** *There is a 36-approximation for the HTSP in metric graphs.*

4 Conclusion

We introduced both the Harmonic Traveling Salesman Problem and the Cumulative Traveling Salesman Problem, in which the cost of traveling an edge depends inversely proportional (or proportional, respectively) on the number of previously visited vertices. We provided a number of new results, including exact solutions in trees with a fixed number of leaves, and approximation for general metric instances.

There are numerous open questions, including the existence of better approximation algorithms, possibly making use of special geometric properties, as well as exact methods for computing provably optimal solutions for benchmark instances of interesting size.

References

1 Hernán Abeledo, Ricardo Fukasawa, Artur Pessoa, and Eduardo Uchoa. The Time Dependent Traveling Salesman Problem: Polyhedra and branch-cut-and-price algorithm. In *Symposium on Experimental Algorithms (SEA)*, pages 202–213, 2010. doi:10.1007/978-3-642-13193-6_18.

2 Tommaso Adamo, Gianpaolo Ghiani, Pierpaolo Greco, and Emanuela Guerriero. Learned upper bounds for the Time Dependent Traveling Salesman Problem. *IEEE Access*, 11:2001–2011, 2023. doi:10.1109/ACCESS.2022.3233852.

3 Ali Fuat Alkaya and Ekrem Duman. A new generalization of the Traveling Salesman Problem. *Applied and Computational Mathematics*, 9(2):162–175, 2010.

4 Aaron Archer and David P. Williamson. Faster approximation algorithms for the Minimum Latency Problem. In *Symposium on Discrete Algorithms (SODA)*, pages 88–96, 2003. URL: http://dl.acm.org/citation.cfm?id=644108.644122.

5 Sanjeev Arora. Polynomial time approximation schemes for Euclidean Traveling Salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998. doi:10.1145/290179.290180.

6 Sanjeev Arora and George Karakostas. Approximation schemes for minimum latency problems. *SIAM Journal on Computing*, 32(5):1317–1337, 2003. doi:10.1137/S0097539701399654.

7 Sanjeev Arora and George Karakostas. A $2 + \varepsilon$ approximation algorithm for the k-MST problem. *Math. Program.*, 107(3):491–504, 2006. doi:10.1007/S10107-005-0693-1.

8 Avrim Blum, Prasad Chalasani, Don Coppersmith, Bill Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. The minimum latency problem. In *Symposium on Theory of Computing (STOC)*, pages 163–171, 1994. doi:10.1145/195058.195125.

9 Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. *SIAM Journal on Computing*, 37(2):653–670, 2007. doi:10.1137/050645464.

10 Kamalika Chaudhuri, Brighten Godfrey, Satish Rao, and Kunal Talwar. Paths, trees, and minimum latency tours. In *Symposium on Foundations of Computer Science (FOCS)*, pages 36–45, 2003. doi:10.1109/SFCS.2003.1238179.

11 Jean-François Cordeau, Gianpaolo Ghiani, and Emanuela Guerriero. Analysis and branch-and-cut algorithm for the Time Dependent Traveling Salesman Problem. *Transportation Science*, 48(1):46–58, 2014. doi:10.1287/trsc.1120.0449.

12 Alfredo Garcıa, Marc Noy, and Javier Tejel. Lower bounds on the number of crossing-free subgraphs of K_N. *Computational Geometry*, 16(4):211–221, 2000. doi:10.1016/S0925-7721(00)00010-9.

13 Naveen Garg. A 3-approximation for the minimum tree spanning k vertices. In *Conference on Foundations of Computer Science (FOCS)*, pages 302–309, 1996. doi:10.1109/SFCS.1996.548489.

14 Naveen Garg. Saving an epsilon: A 2-approximation for the k-MST problem in graphs. In *Symposium on Theory of Computing (STOC)*, page 396–402, 2005. doi:10.1145/1060590.1060650.

15 Michel Goemans and Jon Kleinberg. An improved approximation ratio for the Minimum Latency Problem. *Mathematical Programming*, 82(1):111–124, 1998. doi:10.1007/BF01585867.

16 Chryssi Malandraki and Mark S. Daskin. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transp. Sci.*, 26(3):185–200, 1992. doi:10.1287/TRSC.26.3.185.

17 Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999. `doi:10.1137/S0097539796309764`.

18 R. Ravi, Ravi Sundaram, Madhav V. Marathe, Daniel J. Rosenkrantz, and S. S. Ravi. Spanning trees—short or small. *SIAM Journal on Discrete Mathematics*, 9(2):178–200, 1996. `doi:10.1137/S0895480194266331`.

19 Micha Sharir, Adam Sheffer, and Emo Welzl. Counting plane graphs: perfect matchings, spanning cycles, and Kasteleyn's technique. In *Symposium on Computational Geometry (SoCG)*, pages 189–198, 2012. `doi:10.1145/2261250.2261277`.

20 Micha Sharir, Adam Sheffer, and Emo Welzl. Counting plane graphs: Perfect matchings, spanning cycles, and Kasteleyn's technique. *Journal of Combinatorial Theory, Series A*, 120(4):777–794, 2013. `doi:10.1016/J.JCTA.2013.01.002`.

21 Russ J. Vander Wiel and Nikolaos V. Sahinidis. An exact solution approach for the Time Dependent Traveling Salesman Problem. *Naval Research Logistics (NRL)*, 43(6):797–820, 1996.

Approximation Algorithms for Lawn Mowing with Obstacles

Sándor P. Fekete[1], Linda Kleist[2], Fabian Kollhoff[1], Chek-Manh Loi[1], and Michael Perk[1]

1 Department of Computer Science, TU Braunschweig
fekete@tu-bs.de, {kollhoff,loi,perk}@ibr.cs.tu-bs.de

2 Department of Computer Science, Universität Potsdam
kleist@cs.uni-potsdam.de

Abstract

We consider a geometric optimization problem that generalizes both the Lawn Mowing Problem of covering all of a given region with a unit-sized cutter and the Milling Problem of additionally not leaving the covered area during coverage: For a given polygonal region P and a set of obstacles $\mathcal{O}$, the *Lawn Mowing Problem with Obstacles* asks for a shortest tour that has Euclidean distance 1 to each point in $P \setminus \mathcal{O}$ and distance at least 1 to every point in $\mathcal{O}$. We present constant factor approximations. For the case where the obstacles are strictly contained in P, we present a 21.5-approximation algorithm and a 6.5-approximation for large obstacles. If the obstacles are additionally well-separated, i.e., at least distance $2 + \pi$ apart, we provide a polynomial time 4.96-approximation algorithm.

1 Introduction

The *Lawn Mowing Problem* (LMP) is a well-studied problem in geometric optimization that occurs in a wide range of appplications, such as sensing, surveillance and manufacturing: For a given region P (the lawn) and a unit-radius disk D (the cutter), find a closest roundtrip of shortest Euclidean length that moves the center of D within distance 1 from every point in P. If in addition, the disk is not allowed to cover any point outside of P, we are dealing with the *Milling Problem* (MP), a natural variant motivated by applications such as cutting a desired shape from a block of material. As generalizations of the *Traveling Salesman Problem* (TSP), both problems are NP-hard, with previous work [2] providing approximation algorithms.

In this paper, we consider a generalization of both problems: In the *Lawn Mowing Problem with Obstacles* (LMPO), we seek a shortest tour of D that covers a given region P without intersecting the interior of a designated set $\mathcal{O}$ of obstacles. We focus on the *enclosed LMPO* (e-LMPO) with convex polygonal obstacles of positive area strictly contained in P and separated by at least a distance of 2 from each other to ensure the existence of a feasible tour. Figure 1 illustrates solutions to the LMP, MP, and the LMPO.

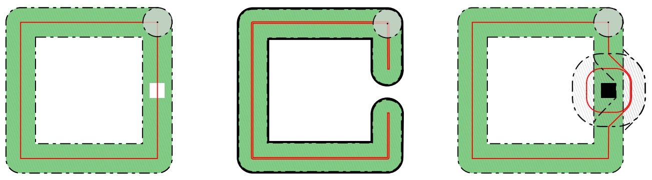


Figure 1 Three examples of a feasible tour for the LMP, MP, and LMPO with a circular cutter.

Our contribution We provide a $(4\pi + 4\sqrt{3} + 2) < 21.5$-approximation algorithm for the e-LMPO that can be improved to 6.46 for instances with large obstacles. For the de-LMPO in which obstacles are *well-separated*, i.e., at least $2+\pi$ apart, we provide a $(2\sqrt{3}\alpha+1.5) < 5$-approximation algorithm, with α being the performance guarantee for a TSP approximation algorithm.

Related work There is a wide range of practical applications for lawn mowing variants, including manufacturing [3, 14, 15], cleaning [7], robotic coverage [8, 9, 13, 16], inspection [12], CAD [11], farming [5, 10, 18] and pest control [6]. The LMP was first introduced by Arkin et al. [1], who later gave the currently best approximation algorithm with a performance guarantee of $2\sqrt{3}\alpha < 3.5\alpha$ [2], where α can be set to $(1+\varepsilon)$ for any $\epsilon > 0$ based on the methods of Arora [4] or Mitchell [17]. The algorithm computes a TSP tour on the dual graph of a hexagonal tiling of the lawn; see Figure 2 (left) for an example.

▶ **Theorem 1.1.** *(Theorem 3 in [2]) The lawn mowing problem has a* $2\sqrt{3}\alpha$*-approximation algorithm.*

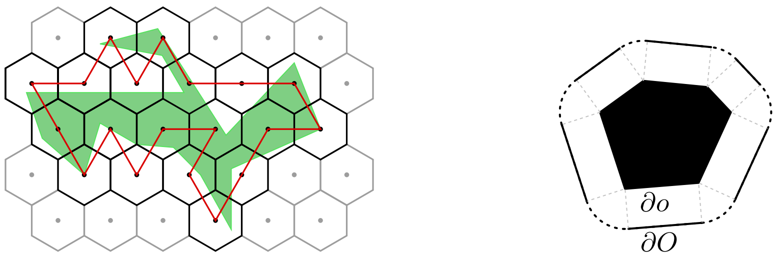


Figure 2 (Left) A hexagonal tiling of the lawn. (Right) The offset boundary ∂O consists of segments and circular arcs. Its total lentgth is given by $|\partial O| = |\partial o| + 2\pi$.

2 e-LMPO approximation

In this section, we present an approximation algorithm for the e-LMPO. For our analysis, we make use of the following simple fact on the offset boundaries of the obstacles; the offset boundary ∂O of an obstacle $o \in \mathcal{O}$ consists of all points at distance 1 of the boundary ∂o of o. For convex obstacles, we have $|\partial O| = |\partial o| + 2\pi$, and we define $\partial\mathcal{O} := \sum_{o\in\mathcal{O}} \partial O$; see Figure 2 (right).

▶ **Lemma 2.1.** *For the e-LMPO, any feasible tour contains the segments of* $\partial\mathcal{O}$*.*

Proof. For an obstacle o, its offset boundary ∂O consists of segments and circular arcs, see Figure 2. For each inner point p of a segment of ∂o, there exists a unique point in ∂O at distance 1. Hence, all segments of $\partial\mathcal{O}$ belong to any feasible tour, see Figure 2. ◀

We now adapt the $2\sqrt{3}\alpha$-approximation algorithm by Arkin et al. [2] to handle obstacles.

▶ **Theorem 2.2.** *The e-LMPO admits an* $(4\pi + 4\sqrt{3} + 2) < 21.5$*-approximation algorithm.*

Proof. For an instance $(P, \mathcal{O})$, the idea is to first cover the boundary of the obstacles and then cover the rest of P using a tiling of the plane with regular hexagons of sidelength 1, see Figure 3. Let $G = (V, E)$ denote the plane graph that has a vertex for each hexagon

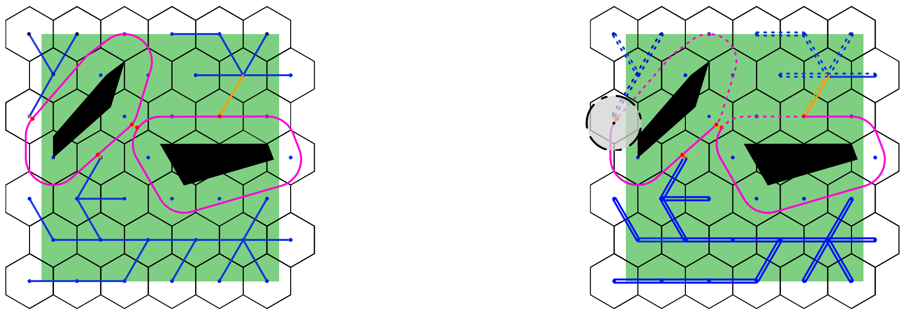


Figure 3 Illustration for the proof of Theorem 2.2. (Left) Spanning trees of the components of $G[V_p - V_o]$ are depicted in blue, offset boundaries in pink and connectors in orange. (Right) The partially traversed tour T is obtained by *walking around* H *and the offset boundary once.*

center and an edge (of length $\sqrt{3}$) between any two hexagons sharing a side. Let $V_p \subset V$ and $V_o \subset V$ denote the sets of vertices whose hexagons intersect $P \setminus \mathcal{O}$ and an obstacle boundary, respectively. We compute a (minimum) spanning tree for each connected component of $G[V_p - V_o]$. We enhance the union of all spanning trees and the offset boundaries to a(n) (abstract) tree H by inserting so-called *connector* edges in a Kruskal-fashion; the length of an edge between $v \in V_p$ (or an ∂O_j) to some ∂O_i is the minimum Euclidean distance between any point of ∂O_i and v (or any point of ∂O_j). Note that each connector has length at most $\sqrt{3}$. Moreover, each obstacle of positive area intersects some hexagon in an interior point. Such a hexagon is not intersected by any other obstacle as they have pairwise distance 2. Consequently, $|\mathcal{O}| \leq |V_o|$. Therefore, we insert at most $|V_p| - |V_o| + |\mathcal{O}| \leq |V_p|$ connectors and H has at most $(2|V_p| - |V_o| - 1)$ edges of length $\sqrt{3}$.

By doubling all edges of H and inserting the offset boundaries as curves, we obtain a Eulerian graph. It contains a tour T of length at most $2(2|V_p| - |V_o|)\sqrt{3} + |\partial\mathcal{O}|$ that visits all vertices $V_p \setminus V_o$ and traverses all offset boundaries of the obstacles, see Figure 3.

By Lemma 2.1, the segments of the offset boundary of an obstacle are contained in any feasible tour. The total length of all segments is $|\partial\mathcal{O}| - |\mathcal{O}|2\pi$. Because any point p in the interior of a segment belongs to at most two offset boundaries, we have $\frac{1}{2}(|\partial\mathcal{O}| - |\mathcal{O}|2\pi) \leq$ OPT; here we use the fact that the obstacles are convex. Together with the fact $|\mathcal{O}| \leq |V_o|$, it follows that

$$|\partial\mathcal{O}| \leq 2\text{OPT} + |\mathcal{O}|2\pi \leq 2\text{OPT} + |V_o|2\pi \tag{1}$$

and hence

$$|T| \leq (4|V_p| - 2|V_o|)\sqrt{3} + |\partial\mathcal{O}| \leq (4|V_p| + 2(\pi/\sqrt{3} - 1)|V_o|)\sqrt{3} + 2\text{OPT}.$$

Note that by disregarding the obstacles, a lawn mowing tour of $P \setminus \mathcal{O}$ is a natural lower bound for an optimal tour in our instance. The tour computed in Theorem 1.1 has length at least $\sqrt{3}|V_p|$ and is a $2\sqrt{3}\alpha$-approximation where α can be arbitrarily close to 1 [4, 17]. This yields an approximation ratio of

$$\frac{|T|}{\text{OPT}} \leq \left(\frac{4\sqrt{3} + 2(\pi/\sqrt{3} - 1)\sqrt{3}}{\sqrt{3}} \cdot 2\sqrt{3} + 2\right) = (4\pi + 4\sqrt{3} + 2). \qquad ◀$$

A better approximation factor can be achieved by restricting the e-LMPO to well-separated obstacles. This allows for better lower bounds.

3 A better approximation for well-separated obstacles

In contrast to the LMP, the presence of obstacles imposes specific structures on the optimal (and any feasible) tour, which can be utilized to establish lower bounds; cf. Lemma 2.1.

3.1 Traversing the boundary of obstacles

Lemma 2.1 motivates the use of the length $|\partial\mathcal{O}|$ as a lower bound for the length of an optimal tour. However, when the obstacles are close to each other, this bound may not hold; see the example in Figure 4. The (black dotted) circular arcs are longer than the connecting (orange) segments. When reducing the height ε of the triangular obstacles, the total length of the circular arcs approaches $\lim_{\varepsilon\to 0} 2\pi$. For obstacles with distance $\delta < \pi$ the total length of the orange segments approaches $\lim_{\varepsilon\to 0} 2\delta < 2\pi$. In the case of e-LMPO, we can show an even better bound; $|\partial\mathcal{O}|$ is a lower bound to the length of an optimal tour if and only if obstacles are *well-separated*, i.e., each pair of obstacles has distance $\geq 2+\pi$. We call this variant de-LMPO.

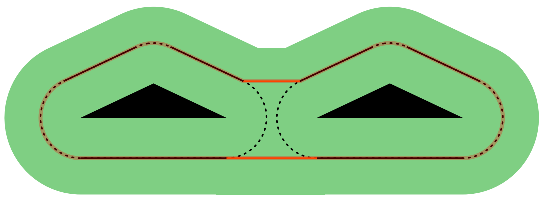


Figure 4 When obstacles are close, then $|\partial\mathcal{O}|$ may not be a lower bound for OPT (in red).

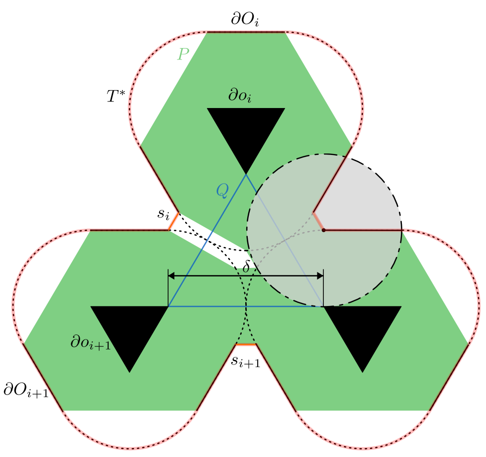
▶ **Theorem 3.1.** *For an instance of de-LMPO with a set of well-separated obstacles $\mathcal{O}$, the optimal solution has length at least $|\partial\mathcal{O}|$. Moreover, the distance bound is best possible, i.e., for each $\varepsilon > 0$, there exists an example where the obstacles have distance at least $2+\pi-\varepsilon$ and the length of the optimal solution is less than $|\partial\mathcal{O}|$.*

Proof. As each obstacle $o \in \mathcal{O}$ is enclosed, its entire boundary ∂o must be visited. The offset boundary ∂O consists of all points of the cutter center that visit ∂o. Because o_i is a convex polygon, ∂O_i consists of segments and circular arcs where each circular arc has length at most π and the total length of the circular arcs sums to 2π; see Figure 2. By Lemma 2.1, all segments of ∂O belong to any feasible tour, which hence has a total length of $|\partial\mathcal{O}| - 2\pi|\mathcal{O}|$. In particular, ∂o is a lower bound.

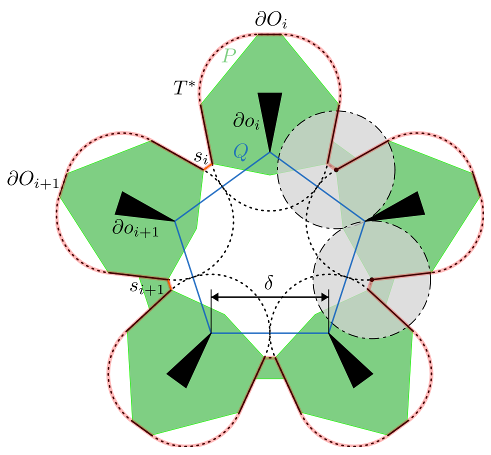
Let T^* be an optimal tour. We call a (maximal) subcurve γ of T^* a *part visiting* $o \in \mathcal{O}$ if its endpoints belong to segments of ∂O and γ contains no point of another $\partial O'$. Furthermore, a *connector* is a subcurve connecting a part visiting o with a part visiting o'. Note that each connector has length at least π. When traversing T^* in some direction, we associate each part visiting o with its proceeding connector. We aim to show that the parts visiting o and their connectors contribute 2π besides the segments contained in ∂O.

If each obstacle o has at least two parts visiting it, then its associated connectors sum to at least 2π. If an obstacle o is visited by just one part, then this part is shortest if it consists of ∂O minus one arc and hence the contribution is at least π (as each arc has a length of at most π). Together with the associated connector, this yields a total contribution of $\geq 2\pi$.

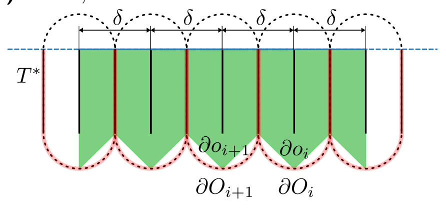
Now, we show that the bound is best possible. Let $\epsilon > 0$, let $\delta := 2+\pi-\varepsilon$, and consider an n-gon Q with side length δ. Each corner of Q is incident to a triangular obstacle, and the lawn P consists of the neighborhood of the obstacle as illustrated in Figure 5.



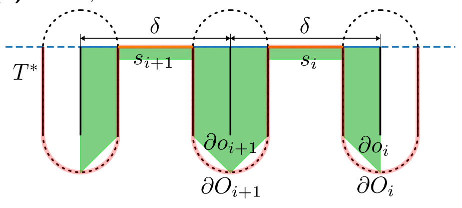
(a) $n = 3$, $\delta = 2$



(b) $n = 5$, $\delta = 2$



(c) For very large n, $\delta = 2$



(d) For very large n, $\delta = 4$

Figure 5 In each example, the polygon P is shaded green, and the red tour T is feasible and has length $|T| < |\partial O|$ if $\delta < 2 + \pi - \epsilon$ for any $\epsilon > 0$.

Except for the inner circular arcs, the optimal tour T^* traverses $\partial\mathcal{O}$ and the connecting segments. The lawn is defined such that T^* covers it. When increasing n and decreasing the width of the obstacles, the unused arc of each offset boundary converges to a length of π, and the length of each connecting segment coverges to $\delta - 2 = \pi - \varepsilon$. Consequently, in the limit, the tour has length $|\partial\mathcal{O}| - n\varepsilon < |\partial\mathcal{O}|$. Thus the bound is best-possible. ◀

3.2 Approximation algorithm for the de-LMPO

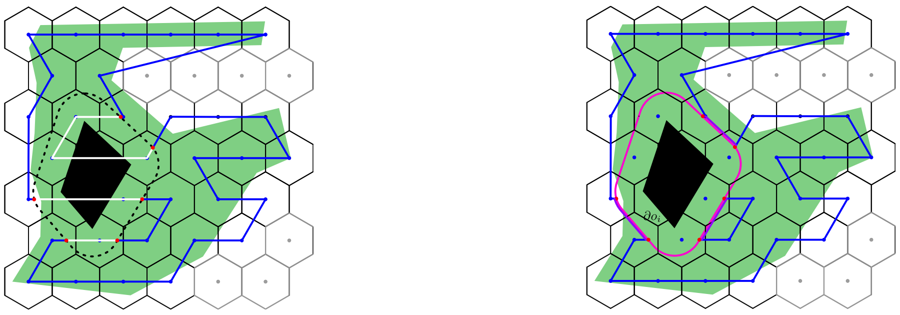
In the de-LMPO variant, all obstacles have distance at least $2 + \pi$ to all other obstacles which allows us to use Theorem 3.1 to obtain a better approximation factor than that of Theorem 2.2.

▶ **Theorem 3.2.** *The de-LMPO has an* $(2\sqrt{3}\alpha + 1.5) < 5$*-approximation algorithm.*

Proof. For a well-seperated instance $(P, \mathcal{O})$, the idea is to cover P using the approximation algorithm from Arkin et al. [2] that uses a tiling of the plane with regular hexagons of sidelength 1 and then introduce detours following $\partial\mathcal{O}$ to cover the lawn around the obstacles. Let $G = (V, E)$ be the plane graph that corresponds to the tiling that has a vertex for each hexagon center and an edge (of length $\sqrt{3}$) between any two hexagons sharing a side. Let $V_p \subset V$ and $V_o \subset V$ denote the set of vertices whose hexagon intersects $P \setminus \mathcal{O}$ and an obstacle boundary, respectively. We compute an α-approximate TSP tour T' that visits all hexagon centers in V_p, where α can be $(1 + \varepsilon)$ (and thus arbitrarily close to 1) based on the methods of [4, 17].

We proceed by removing parts of T' that lie in the offset region of the obstacles $\mathcal{O}$ and obtain a set of disconnected paths $\{\pi_1, \pi_2, \dots\}$; see Figure 6a. Each path π_i =

$(v_1, v_2, \dots, v_{n_i-1}, v_{n_i})$ contains points $v_2, \dots, v_{n_i-1} \in V_p \setminus V_o$ and intersects $\partial\mathcal{O}$ in its endpoints v_1, v_{n_i}. Let $k_i \geq 0$ be the number of endpoints that lie on the offset boundary of an obstacle o_i. We call the union of all endpoints the *connection points* V_c with $|V_c| = \sum_{o_i \in \mathcal{O}} k_i$.



(a) TSP approximation on V_p.

(b) Eulerian graph H'.

Figure 6 de-LMPO approximation with blue TSP tour, pink graph H and Eulerian graph H'.

Consider the graph H with vertices $(V_p \setminus V_o) \cup V_c$ and edges according to the paths $\pi_1, \dots, \pi_k$ that is further enhanced by adding edges between the connection points on the offset boundaries of the obstacles. We order the k_i connection points on each offset boundary ∂O_i in counterclockwise order and connect them via edges that follow ∂O_i. By Theorem 3.1, the total length of the newly added edges is $|\partial\mathcal{O}| \leq \text{OPT}$. Adding a second copy of every second edge around each offset boundary ensures that every connection point has an even degree, see Figure 6b. The last step can be done by inserting edge of total length at most $\frac{1}{2}|\partial\mathcal{O}| \leq \frac{1}{2}\text{OPT}$. The resulting Eulerian graph H' contains a feasible tour T that traverses all offset boundaries $\partial\mathcal{O}$ and visits all vertices in $V_p \setminus V_o$ as well as all connection points V_c.

By Theorems 1.1 and 3.1 the edges in H cost at most $2\sqrt{3}\alpha\text{OPT}$ and the additional edges in H' cost at most 1.5OPT. Thus, in the worst case, $|T'| \leq (2\sqrt{3}\alpha + 1.5)\text{OPT}$. ◀

4 Approximation for large obstacles

In some practical applications, the perimeter of the obstacles is large compared to the cutter. This motivates e-LMPO[ρ] where each obstacle has perimeter at least ρ. For e-LMPO[ρ], we can bound $|\partial\mathcal{O}|$ by inserting ρ into Equation (1), which yields $|\partial\mathcal{O}| \leq 2\left(1 + \frac{\pi}{\rho}\right)\text{OPT}$. Using this bound, we modify the analysis of the algorithm from Theorem 3.2 from $1.5|\partial\mathcal{O}| \leq 1.5\text{OPT}$ to $1.5|\partial\mathcal{O}| \leq 3\left(1 + \frac{\pi}{\rho}\right)\text{OPT}$.

▶ **Corollary 4.1.** *The e-LMPO[ρ] has an $\left(2\sqrt{3}\alpha + 3\left(1 + \frac{\pi}{\rho}\right)\right)$-approximation algorithm.*

For large ρ, the approximation factor converges to $2\sqrt{3}\alpha + 3 < 6.5$.

5 Conclusion

We introduced the e-LMPO and provided a < 21.5-approximation algorithm in Section 2. For the de-LMPO with obstacles at least $2 + \pi$ apart, we achieved a < 5-approximation algorithm. A new analysis of the first algorithm also leads to a 6.46-approximation for large obstacles. Several open questions remain, such as algorithms for LMPO with arbitrary obstacles (not necessarily convex or inside P) or the existence of a PTAS. Better lower bounds for any variant could lead to improved approximations and exact algorithms.

References

1 Esther M. Arkin, Sándor P. Fekete, and Joseph S. B. Mitchell. The lawnmower problem. In *Canadian Conference on Computational Geometry (CCCG)*, pages 461–466, 1993. URL: `https://cglab.ca/~cccg/proceedings/1993/Paper79.pdf`.

2 Esther M. Arkin, Sándor P. Fekete, and Joseph S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17:25–50, 2000. `doi:10.1016/S0925-7721(00)00015-8`.

3 Esther M. Arkin, Martin Held, and Christopher L. Smith. Optimization problems related to zigzag pocket machining. *Algorithmica*, 26(2):197–236, 2000. `doi:10.1007/s004539910010`.

4 Sanjeev Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998. `doi:10.1145/290179.290180`.

5 Rik Bähnemann, Nicholas Lawrance, Jen Jen Chung, Michael Pantic, Roland Siegwart, and Juan Nieto. Revisiting Boustrophedon coverage path planning as a generalized traveling salesman problem. In *Field and Service Robotics*, pages 277–290, 2021. `doi:10.1007/978-981-15-9460-1_20`.

6 Aaron T. Becker, Mustapha Debboun, Sándor P. Fekete, Dominik Krupke, and An Nguyen. Zapping Zika with a mosquito-managing drone: Computing optimal flight patterns with minimum turn cost. In *Symposium on Computational Geometry (SoCG)*, pages 62:1–62:5, 2017. `doi:10.4230/LIPIcs.SoCG.2017.62`.

7 Richard Bormann, Joshua Hampp, and Martin Hägele. New brooms sweep clean–An autonomous robotic cleaning assistant for professional office cleaning. In *International Conference on Robotics and Automation (ICRA)*, pages 4470–4477, 2015. `doi:10.1109/ICRA.2015.7139818`.

8 Tauã M. Cabreira, Lisane B. Brisolara, and Paulo R. Ferreira Jr. Survey on coverage path planning with unmanned aerial vehicles. *Drones*, 3(1):4, 2019. `doi:10.3390/drones3010004`.

9 Howie Choset. Coverage for robotics–a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1):113–126, 2001. `doi:10.1023/A:1016639210559`.

10 Howie Choset and Philippe Pignon. Coverage path planning: The Boustrophedon cellular decomposition. In *Field and Service Robotics*, pages 203–209, 1998. `doi:10.1007/978-1-4471-1273-0_32`.

11 Gershon Elber and Myung-Soo Kim. Offsets, sweeps and Minkowski sums. *Computer-Aided Design*, 31(3), 1999. `doi:10.1016/S0010-4485(99)00012-3`.

12 Brendan Englot and Franz Hover. Sampling-based coverage path planning for inspection of complex structures. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 29–37, 2012. URL: `http://www.aaai.org/ocs/index.php/ICAPS/ICAPS12/paper/view/4728`.

13 Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013. `doi:10.1016/j.robot.2013.09.004`.

14 Martin Held. *On the Computational Geometry of Pocket Machining*, volume 500 of *LNCS*. Springer, 1991. `doi:10.1007/3-540-54103-9`.

15 Martin Held, Gábor Lukács, and László Andor. Pocket machining based on contour-parallel tool paths generated by means of proximity maps. *Computer-Aided Design*, 26(3):189–203, 1994. `doi:10.1016/0010-4485(94)90042-6`.

16 Katharin R. Jensen-Nau, Tucker Hermans, and Kam K. Leang. Near-optimal area-coverage path planning of energy-constrained aerial robots with application in autonomous environmental monitoring. *IEEE Transactions on Automation Science and Engineering*, 18(3):1453–1468, 2021. doi:10.1109/TASE.2020.3016276.

17 Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999. doi:10.1137/S0097539796309764.

18 Timo Oksanen and Arto Visala. Coverage path planning algorithms for agricultural field machines. *Journal of Field Robotics*, 26(8):651–668, 2009. doi:10.1002/rob.20300.

Moving Matter: Efficient Reconfiguration of Tile Arrangements by a Single Active Robot*

Aaron T. Becker[1], Sándor P. Fekete[2], Jonas Friemel[3], Ramin Kosfeld[2], Peter Kramer[2], Harm Kube[4], Christian Rieck[5], Christian Scheffer[3], and Arne Schmidt[2]

1 **Electrical Engineering, University of Houston, Texas, USA**
`atbecker@uh.edu`

2 **Computer Science, TU Braunschweig, Braunschweig, Germany**
`s.fekete@tu-bs.de, {kosfeld, kramer, aschmidt}@ibr.cs.tu-bs.de`

3 **Electrical Engineering and Computer Science, Bochum University of Applied Sciences, Bochum, Germany**
`{jonas.friemel, christian.scheffer}@hs-bochum.de`

4 **Computer Science, TU Berlin, Berlin, Germany**
`h.kube@campus.tu-berlin.de`

5 **Discrete Mathematics, University of Kassel, Kassel, Germany**
`christian.rieck@mathematik.uni-kassel.de`

Abstract

We consider the problem of reconfiguring a two-dimensional connected grid arrangement of passive building blocks from a start configuration to a goal configuration, using a single active robot that can move on the tiles, remove individual tiles from a given location and physically move them to a new position by walking on the remaining configuration. The objective is to determine a reconfiguration schedule that minimizes the overall makespan, while ensuring that the tile configuration remains connected. We provide the following: (1) We present a generalized version of the problem, parameterized by weighted costs for moving with or without tiles, and show that this is NP-hard. (2) We give a polynomial-time constant-factor approximation for the case of disjoint start and target bounding boxes. Our algorithm yields optimal carry distance for 2-scaled instances.

Related Version `arXiv:2502.09299`

1 Introduction

Building and modifying structures consisting of many basic components is an important objective, both in fundamental theory and in a spectrum of practical settings. Transforming such structures with the help of autonomous robots is particularly relevant in very large [9] and very small dimensions [25] that are hard to access for direct human manipulation, e.g., in extraterrestrial space [7] or in microscopic environments [5].

Progress in material sciences has spawned discrete, light-weight materials that allow building large lattice structures of tiles that can be manipulated by simple robots for reconfiguration [21], as shown in Figure 1: The robot can move on the tile arrangement, remove individual tiles and physically relocate them to a new position by walking on the remaining configuration, which needs to remain connected at all times.

* Work from the University of Houston was partially supported by NSF grant IIS-2130793. Work from TU Braunschweig and HS Bochum was partially supported by the German Research Foundation (DFG), project “Space Ants”, FE 407/22-1 and SCHE 1931/4-1. Work from the University of Kassel was partially supported by DFG grant 522790373.

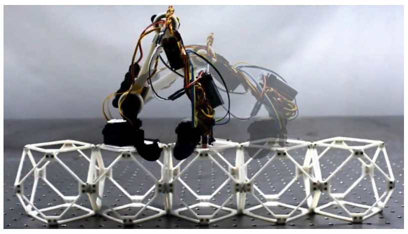


Figure 1 A simple Bill-E robot that can move on a configuration of digital, light-weight material and relocate individual voxels for overall reconfiguration. Photos adapted from [21].

How can we use such a robot to transform a given start configuration into a desired goal arrangement, as quickly as possible? We provide the following results.

1. We present a generalized version of the problem, parameterized by weighted costs for moving with or without tiles, and show that this is NP-hard.
2. We give a polynomial-time constant-factor approximation in case of disjoint start and target bounding boxes. Our approach yields optimal carry distance for 2-scaled instances.

Due to limited space, details for statements marked by ($\star$) can be found in the full version [6]. However, we provide a concise overview of the ideas.

Related work. Garcia et al. [15, 16] showed that computing optimal schedules for robotic reconfiguration is NP-hard. They designed heuristic approaches exploiting rapidly exploring random trees (RRT), and a time-dependent variant of the A* search algorithm.

A different context for reconfiguration arises from programmable matter [17, 18, 19]. Here, even finite automata are capable of building bounding boxes from tiles around polyominoes, as well as scale and rotate them while maintaining connectivity at all times [12, 24].

For arrangements that are composed of active, self-moving objects (or agents), a number of related results have been obtained. For the *sliding cube model* [13, 14], Akitaya et al. [3] show that universal sequential reconfiguration in two dimensions is possible, even while maintaining connectivity of all intermediate configurations, but minimizing the makespan of a schedule is NP-complete. Abel et al. [1] and Kostitsyna et al. [22] gave similar results in three dimensions. Most recently, Akitaya et al. [4] give results for the *parallel sliding square model.* In a related model, Fekete et al. [10, 11] show that parallel connected reconfiguration of a swarm of (labeled) agents is NP-complete, even for deciding whether there is a schedule of makespan 2, and present algorithms for computing constant stretch schedules, i.e., the ratio between the makespan of a schedule and a natural lower bound (the maximum minimum distance between an individual start and target position) is bounded by a constant.

Preliminaries. For the following, we refer to Figure 2. We are given a fixed set of n indistinguishable square *tile* modules located at discrete, unique *positions* in the infinite integer grid $\mathbb{Z}^2$. If this set induces a connected subgraph, where two positions are considered connected if either their x- or y-coordinate differs by 1, we say that the tiles form a connected *configuration* or *polyomino.* Let $\mathcal{C}(n)$ refer to the set of all polyominoes of n tiles.

Consider a *robot* that occupies a single tile at any given time and uses cardinal directions to navigate; the unit vectors $(1, 0)$ and $(0, 1)$ correspond to *east* and *north*, respectively. In discrete time steps, the robot can either move to an adjacent tile, pick up an adjacent tile (if it is not carrying one), or place a tile at an adjacent unoccupied position (if it is carrying a tile). A tile may only be picked up if the configuration remains connected without it.

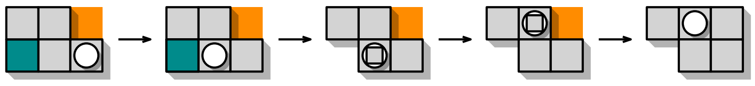


Figure 2 An example schedule for some $C_s \rightrightarrows C_t$: The robot moves in cardinal directions, walking on and modifying tiles. Tiles in $C_s \cap C_t$ are marked in gray, $C_s \setminus C_t$ in cyan, and $C_t \setminus C_s$ in orange.

Given two connected configurations C_s and C_t, a *(reconfiguration) schedule* S is a finite, connectivity-preserving sequence of operations to be performed by the robot for $C_s \rightrightarrows C_t$ exactly if it transforms C_s into C_t. Let $d_C(S)$ denote the *carry distance*, which is the sum of geodesic distances between consecutive pickups and drop-offs in S. This represents the total distance the robot travels while carrying a tile, with an additional unit of distance added each time the robot either picks up or places a tile. Similarly, the *empty distance* $d_E(S)$ is the geodesic distance walked without carrying a tile.

In this paper, we consider the Single Robot Reconfiguration problem: Given two connected configurations C_s and C_t, and a *rational weight factor* $\lambda \in [0,1]$, our goal is to compute a schedule S for $C_s \rightrightarrows C_t$ of minimum *makespan* $|S| := \lambda \cdot d_E(S) + d_C(S)$. The associated decision problem is defined as expected with an upper bound on the length of a schedule. We refer to the minimum weighted makespan for a given instance as OPT.

2 Computational complexity of the problem

We start by investigating the computational complexity of the decision variant of the generalized reconfiguration problem. In particular, we prove that the problem is NP-hard for any rational factor $\lambda \in [0,1]$. This generalizes a result by Garcia et al. [16] for $\lambda = 1$.

▶ **Theorem 2.1** ($\star$). *Single Robot Reconfiguration is* NP*-hard, parameterized by* λ.

We distinguish between two cases. If $\lambda \in (0,1]$, we reduce from the Hamiltonian path problem in induced subgraphs of the infinite grid graph [20]. The high-level idea is to expand the grid graph of a given Hamiltonian path instance, placing small reconfiguration tasks at each vertex of the graph.

For $\lambda = 0$, we reduce from Planar Monotone 3Sat [8]. We utilize and adapt the reduction given by Akitaya et al. [3] for the sequential sliding squares problem. As $\lambda = 0$, the robot is effectively allowed to "teleport" across the configuration. Therefore, we construct variable and clause gadgets in a way that a unit square must be carried through one side of a variable in order to perform a constant number of reconfiguration steps within the clauses.

3 Constant-factor approximation

We now turn to a special case of the optimization variant in which the configurations have *disjoint bounding boxes*, i.e., there exists an axis-parallel bisector that separates them. Let this bisector be horizontal such that the target configuration lies south. We present an algorithm that computes schedules of makespan at most $c \cdot \text{OPT}$ for some fixed $c \geq 1$.

3.1 2-scaled instances

We additionally impose the constraint that both the start and target configurations are 2-*scaled*, i.e., they consist of 2×2-squares of tiles aligned with a 2×2 integer grid, and show:

▶ **Theorem 3.1** (⋆). *There exists a constant c such that for any pair of 2-scaled configurations $C_s, C_t \in \mathcal{C}(n)$ with disjoint bounding boxes, we can efficiently compute a schedule for $C_s \rightrightarrows C_t$ with weighted makespan at most $c \cdot OPT$.*

Our algorithm utilizes a type of intermediate configuration called *histogram*. A histogram consists of a *base* strip of unit height and (multiple) orthogonal unit width *columns* attached to its base. The direction of its columns determines the orientation of a histogram, e.g., the histogram H_s in Figure 3 is *north-facing*. We proceed in three phases, see Figure 3.

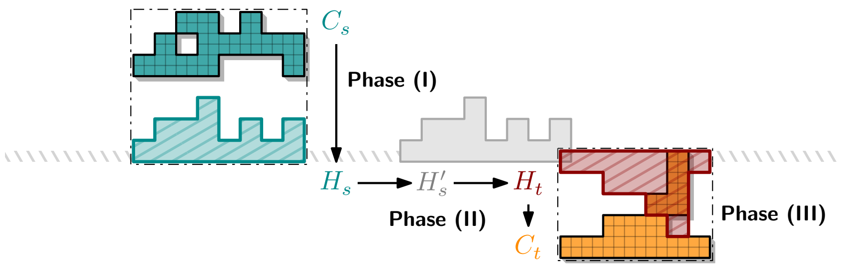


Figure 3 An example for a start and target configuration C_s and C_t, the intermediate histograms H_s and H_t sharing a baseline, and the horizontally translated H'_s that shares a tile with H_t.

Phase (I). Transform the configuration C_s into a north-facing histogram H_s.
Phase (II). Translate H_s to overlap with the target bounding box and transform it into a south-facing histogram H_t contained in the bounding box of C_t.
Phase (III). Finally, apply **Phase (I)** in reverse to obtain C_t from H_t.

Since **Phases (I)** and **(III)** are largely identical, we reduce to two subroutines: Transforming a 2-scaled configuration into a 2-scaled histogram and converting two such histograms into one another. We denote the optimal carry distance for any schedule $C_s \rightrightarrows C_t$ by $\sigma(C_s, C_t)$.

Phase (I): Transform a configuration into a histogram. We proceed by describing a subroutine that constructs a histogram from an arbitrary 2-scaled configuration by moving tiles strictly in one cardinal direction. The resulting histogram faces the opposite direction.

▶ **Lemma 3.2** (⋆). *Let C_s be a 2-scaled polyomino and let H_s be a histogram that can be created from C_s by moving tiles in only one cardinal direction. We can efficiently compute a schedule of makespan $\mathcal{O}(n + \sigma(C_s, H_s))$ for $C_s \rightrightarrows H_s$ with optimal carry distance.*

Our strategy is simply as follows: We iteratively move sets of tiles by two units into the respective target direction, until the histogram is constructed. We give a high-level explanation of our approach by example of a north-facing histogram, as depicted in Figure 4.

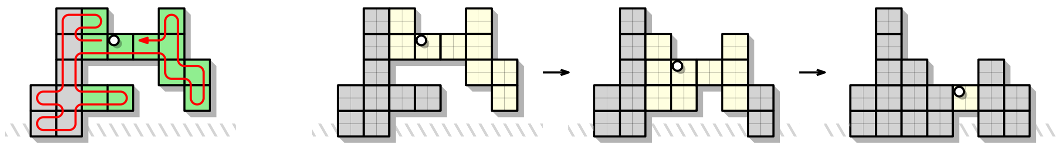


Figure 4 Left: A walk across all tiles (red), the set H (gray) and two free components (green). Right: Based on the walk, the free components are iteratively moved south to reach a histogram shape. The free component that is going to be translated south next is highlighted in yellow.

Let P be any intermediate 2-scaled polyomino obtained by moving tiles south while realizing $C_s \rightrightarrows H_s$. Let H be the set of maximal vertical strips of tiles that contain a base tile in H_s, i.e., all tiles that do not need to be moved further south. We define the *free components* of P as the set of connected components in $P \setminus H$. By definition, once a tile becomes part of H, it is not moved again until the target histogram H_s is obtained.

The robot now simply walks across the entire polyomino, and whenever it enters a free component, it performs a subroutine that translates the component south by two units by repeatedly moving the northernmost tile in a column south. As the configuration is 2-scaled, this guarantees connectivity; we refer to Figure 4 for an illustration.

▶ **Lemma 3.3** ($\star$). *Given a free component F of a 2-scaled polyomino P, we can efficiently compute a schedule of makespan $\mathcal{O}(|F|)$ to translate F in the target direction by two units.*

By applying Lemma 3.3 on the whole polyomino P instead of just a free component, we can translate P in any direction with asymptotically optimal makespan.

▶ **Corollary 3.4.** *Any 2-scaled polyomino can be translated by k units in any cardinal direction by a schedule of weighted makespan $\mathcal{O}(n \cdot k)$.*

Phase (II): Reconfigure a histogram into a histogram. By the assumption of the existence of a horizontal bisector between the bounding boxes of C_s and C_t, the histogram H_s is north-facing, whereas H_t is south-facing. The bounding box of C_s is vertically extended to share exactly one y-coordinate with the bounding box of C_t, and this is where both histogram bases are placed; see Figure 3. By Corollary 3.4, the tiles in H_s can be moved toward H_t in asymptotically optimal makespan until the histogram bases share a tile.

▶ **Lemma 3.5** ($\star$). *Let H_s be a north-facing and H_t a south-facing histogram that share at least one base tile. We can efficiently compute a schedule of makespan $\mathcal{O}(n + \sigma(H_s, H_t))$ for $H_s \rightrightarrows H_t$ with optimal carry distance.*

Figure 5 illustrates our approach: We iteratively move the northernmost westernmost tile of H_s to the northernmost westernmost unoccupied position in H_t until H_t is constructed. That position may not be reachable initially, in which case we first extend the histogram base in western direction. This ordering ensures that tiles are moved on shortest paths to H_t.

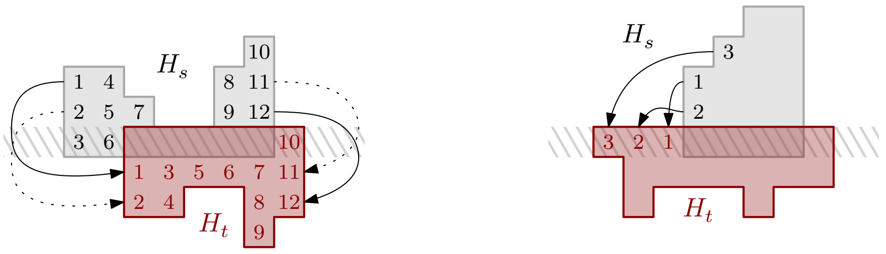


Figure 5 Left: Ordering of tile moves for $H_s \rightrightarrows H_t$. Right: If the westernmost unoccupied position in H_t is unreachable, the base may need to be extended first.

By Lemmas 3.2 and 3.5, tiles are moved with optimal carry distance in **Phases (I)**, **(II)**, and **(III)**. We can show that the combined paths remain shortest possible, yielding an optimal schedule for $\lambda = 0$, i.e., if there is no movement cost when the robot is not carrying a tile.

▶ **Corollary 3.6.** *For any pair of 2-scaled configurations $C_s, C_t \in \mathcal{C}(n)$ with disjoint bounding boxes and $\lambda = 0$, we can efficiently compute an optimal schedule for $C_s \rightrightarrows C_t$.*

3.2 General instances

The key advantage of 2-scaled instances is the absence of cut vertices, which simplifies the maintenance of connectivity during the reconfiguration, which we now tackle separately.

Most parts of our previous method already work independent of the configuration scale. The only required modification concerns the translation of free components, as the polyomino may become disconnected while moving free components that are not 2-scaled. The key technique here is that two auxiliary tiles can be used to preserve connectivity at cut vertices that need to be moved; an example is illustrated in Figure 6. The use of auxiliary tiles to preserve connectivity is also exploited in other models [2, 23]. For this, we can use any two non-cut vertex tiles from the starting configuration, e.g., any leaf from a spanning tree of C_s.

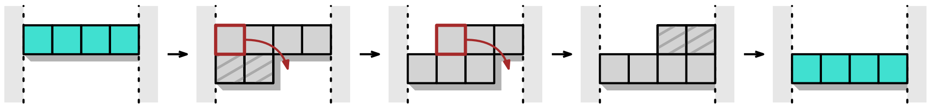


Figure 6 Translating a *corridor* (in cyan) of width 4 south, using two auxiliary (hatched) tiles.

The high level idea of the adjustment is as follows: We decompose each free component F into its *elements*; (1) maximal vertical *strips* of unit width and (2) maximal horizontal *corridors* of unit height. As translating a single element may cause disconnection to adjacent elements, we apply a recursive strategy that handles the elements *blocking* our translation, i.e., that would yield a disconnected configuration, first. We then move the current element, and finally process all other adjacent elements. Finally, we obtain the following.

▶ **Theorem 3.7** ($\star$). *There exists a constant c such that for any pair of configurations $C_s, C_t \in \mathcal{C}(n)$ with disjoint bounding boxes, we can efficiently compute a schedule for $C_s \rightrightarrows C_t$ with weighted makespan at most $c \cdot OPT$.*

4 Conclusions and future work

Our paper presents progress on the reconfiguration problem for tile-based structures with a single active robot. In particular, we showed that the problem is NP-hard for any weighted cost function based on walking and carrying. Complementarily, we developed an constant-factor approximation algorithm to reconfigure two polyominoes into one another in the case that both configurations are contained in disjoint bounding boxes.

Several open questions remain: It seems plausible that our methods can be generalized to be performed by many robots in parallel. Much more intricate is the question on whether a fully distributed approach is possible. Finally, can we adapt our approach to instances in which the bounding boxes of the configurations are intersecting, i.e., overlapping or nested?

References

1 Zachary Abel, Hugo A. Akitaya, Scott Duke Kominers, Matias Korman, and Frederick Stock. A universal in-place reconfiguration algorithm for sliding cube-shaped robots in a quadratic number of moves. In *Symposium on Computational Geometry (SoCG)*, pages 1:1–1:14, 2024. doi:10.4230/LIPICS.SOCG.2024.1.

2 Hugo A. Akitaya, Esther M. Arkin, Mirela Damian, Erik D. Demaine, Vida Dujmovic, Robin Y. Flatland, Matias Korman, Belén Palop, Irene Parada, André van Renssen, and Vera Sacristán. Universal reconfiguration of facet-connected modular robots by pivots: The $O(1)$ musketeers. *Algorithmica*, 83:1316–1351, 2021. doi:10.1007/s00453-020-00784-6.

3 Hugo A. Akitaya, Erik D. Demaine, Matias Korman, Irina Kostitsyna, Irene Parada, Willem Sonke, Bettina Speckmann, Ryuhei Uehara, and Jules Wulms. Compacting squares: Input-sensitive in-place reconfiguration of sliding squares. In *Scandinavian Symposium on Algorithm Theory (SWAT)*, pages 4:1–4:19, 2022. doi:10.4230/LIPICS.SWAT.2022.4.

4 Hugo A. Akitaya, Sándor P. Fekete, Peter Kramer, Saba Molaei, Christian Rieck, Frederick Stock, and Tobias Wallner. Sliding squares in parallel, 2024. doi:10.48550/arXiv.2412.05523.

5 Aaron T. Becker, Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, Christian Rieck, Christian Scheffer, and Arne Schmidt. Tilt assembly: Algorithms for micro-factories that build objects with uniform external forces. *Algorithmica*, 82(2):165–187, 2020. doi:10.1007/S00453-018-0483-9.

6 Aaron T. Becker, Sándor P. Fekete, Jonas Friemel, Ramin Kosfeld, Peter Kramer, Harm Kube, Christian Rieck, Christian Scheffer, and Arne Schmidt. Moving matter: Efficient reconfiguration of tile arrangements by a single active robot, 2025. doi:10.48550/arXiv.2502.09299.

7 Mohamed Khalil Ben-Larbi, Kattia Flores Pozo, Tom Haylok, Mirue Choi, Benjamin Grzesik, Andreas Haas, Dominik Krupke, Harald Konstanski, Volker Schaus, Sándor P. Fekete, Christian Schurig, and Enrico Stoll. Towards the automated operations of large distributed satellite systems. Part 1: Review and paradigm shifts. *Advances in Space Research*, 67(11):3598–3619, 2021. doi:10.1016/j.asr.2020.08.009.

8 Mark de Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *International Journal on Computational Geometry and Applications*, 22(3):187–206, 2012. doi:10.1142/S0218195912500045.

9 Sándor P. Fekete. Coordinating swarms of objects at extreme dimensions. In *International Workshop on Combinatorial Algorithms (IWOCA)*, pages 3–13, 2020. doi:10.1007/978-3-030-48966-3_1.

10 Sándor P. Fekete, Phillip Keldenich, Ramin Kosfeld, Christian Rieck, and Christian Scheffer. Connected coordinated motion planning with bounded stretch. *Autonomous Agents and Multi-Agent Systems*, 37(2), 2023. doi:10.1007/S10458-023-09626-5.

11 Sándor P. Fekete, Peter Kramer, Christian Rieck, Christian Scheffer, and Arne Schmidt. Efficiently reconfiguring a connected swarm of labeled robots. *Autonomous Agents and Multi-Agent Systems*, 38(2), 2024. doi:10.1007/s10458-024-09668-3.

12 Sándor P. Fekete, Eike Niehs, Christian Scheffer, and Arne Schmidt. Connected reconfiguration of lattice-based cellular structures by finite-memory robots. *Algorithmica*, 84(10):2954–2986, 2022. doi:10.1007/s00453-022-00995-z.

13 Robert Fitch, Zack J. Butler, and Daniela Rus. Reconfiguration planning for heterogeneous self-reconfiguring robots. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2460–2467, 2003. doi:10.1109/IROS.2003.1249239.

14 Robert Fitch, Zack J. Butler, and Daniela Rus. Reconfiguration planning among obstacles for heterogeneous self-reconfiguring robots. In *International Conference on Robotics and Automation (ICRA)*, pages 117–124, 2005. doi:10.1109/ROBOT.2005.1570106.

15 Javier Garcia, Michael Yannuzzi, Peter Kramer, Christian Rieck, and Aaron T. Becker. Connected reconfiguration of polyominoes amid obstacles using RRT*. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 6554–6560, 2022. doi:10.1109/IROS47612.2022.9981184.

16 Javier Garcia, Michael Yannuzzi, Peter Kramer, Christian Rieck, Sándor P. Fekete, and Aaron T. Becker. Reconfiguration of a 2D structure using spatio-temporal planning and load transferring. In *International Conference on Robotics and Automation (ICRA)*, pages 8735–8741, 2024. doi:10.1109/ICRA57147.2024.10611057.

17 Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Fabian Kuhn, Dorian Rudolph, and Christian Scheideler. Shape recognition by a finite automaton robot. In *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 52:1–52:15, 2018. doi:10.4230/LIPIcs.MFCS.2018.52.

18 Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Fabian Kuhn, Dorian Rudolph, Christian Scheideler, and Thim Strothmann. Forming tile shapes with simple robots. *Natural Computing*, 19(2):375–390, 2020. doi:10.1007/s11047-019-09774-2.

19 Kristian Hinnenthal, David Liedtke, and Christian Scheideler. Efficient shape formation by 3D hybrid programmable matter: An algorithm for low diameter intermediate structures. In *Symposium on Algorithmic Foundations of Dynamic Networks (SAND)*, pages 15:1–15:20, 2024. doi:10.4230/LIPICS.SAND.2024.15.

20 Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982. doi:10.1137/0211056.

21 Benjamin Jenett, Amira Abdel-Rahman, Kenneth Cheung, and Neil Gershenfeld. Material–robot system for assembly of discrete cellular structures. *IEEE Robotics and Automation Letters*, 4(4):4019–4026, 2019. doi:10.1109/LRA.2019.2930486.

22 Irina Kostitsyna, Tim Ophelders, Irene Parada, Tom Peters, Willem Sonke, and Bettina Speckmann. Optimal in-place compaction of sliding cubes. In *Scandinavian Symposium on Algorithm Theory (SWAT)*, pages 31:1–31:14, 2024. doi:10.4230/LIPICS.SWAT.2024.31.

23 Othon Michail, George Skretas, and Paul G. Spirakis. On the transformation capability of feasible mechanisms for programmable matter. *Journal of Computer and System Sciences*, 102:18–39, 2019. doi:10.1016/j.jcss.2018.12.001.

24 Eike Niehs, Arne Schmidt, Christian Scheffer, Daniel E. Biediger, Michael Yannuzzi, Benjamin Jenett, Amira Abdel-Rahman, Kenneth C. Cheung, Aaron T. Becker, and Sándor P. Fekete. Recognition and reconfiguration of lattice-based cellular structures by simple robots. In *International Conference on Robotics and Automation (ICRA)*, pages 8252–8259, 2020. doi:10.1109/ICRA40945.2020.9196700.

25 Yangsheng Zhu, Mingjing Qi, Zhiwei Liu, Jianmei Huang, Dawei Huang, Xiaojun Yan, and Liwei Lin. A 5-mm untethered crawling robot via self-excited electrostatic vibration. *IEEE Transactions on Robotics*, 38(2):719–730, 2022. doi:10.1109/TRO.2021.3088053.

Layered Graph Drawing with Few Gaps and Few Crossings

Alexander Dobler[1] **and Jakob Roithinger**[1]

1 **Algorithms and Complexity Group, TU Wien**
adobler@ac.tuwien.ac.at,jakob.roithinger@student.tuwien.ac.at

Abstract

We consider the task of drawing a graph on multiple horizontal layers, where each node is assigned a layer, and each edge connects nodes of different layers. Known algorithms determine the orders of nodes on each layer to minimize crossings between edges, increasing readability. Usually, this is done by repeated one-sided crossing minimization for each layer. These algorithms allow edges that connect nodes on non-neighboring layers, called "long" edges, to weave freely throughout layers of the graph, creating many "gaps" in each layer. As shown in a recent work on hive plots – a similar visualization drawing vertices on multiple layers – it can be beneficial to restrict the number of such gaps. We extend existing heuristics and exact algorithms for one-sided crossing minimization in a way that restricts the number of allowed gaps. The extended heuristics maintain approximation ratios, and in an experimental evaluation we show that they perform well with respect to the number of resulting crossings when compared with exact ILP formulations.

1 Introduction

Drawing graphs is a non-trivial task, and many visualization approaches exist. One such approach, known as *layered graph drawing*, draws the nodes on horizontal layers $L = \{L_1, L_2, \ldots, L_\ell\}$, each edge connects nodes of different layers. Sugiyama et al. pioneered the automation of such drawings [12] in the well-known *Sugiyama-framework* consisting of multiple steps. The first step assigns nodes to the ℓ layers such that nodes connected by an edge are on different layers (Figure 1a). In the next step so-called *long edges* connecting nodes u and v of non-neighbouring layers L_i and L_j, $i < j - 1$, are replaced by a path of length $j - i$. The newly created *dummy* nodes are assigned to layers $i + 1, i + 2, \ldots, j - 1$ (Figure 1b). Original nodes are called *real nodes*. After this process each edge connects nodes of adjacent layers. In the next step edge crossings are reduced by permuting the nodes of each layer. Usually, this is performed for neighboring layers, whereby the order of nodes in one layer is fixed and the other layer is permuted. This is known as one-sided crossing minimization (OSCM) [4]. OSCM is performed iteratively, "up" and "down" the layers of the graph, i.e. for $i = 1, 2, \ldots, \ell - 1$ layer L_i is fixed and layer L_{i+1} is permuted. Then, for $i = \ell, \ell - 1, \ldots, 2$, layer L_i is fixed and layer L_{i-1} is permuted. Several such "up" and "down" runs may be performed until reaching a termination condition. The last step replaces dummy nodes by the original edges, and assigns x-coordinates to nodes.

We are concerned with the second step of the above framework. In existing algorithms, dummy and real nodes are treated equally during the crossing minimization step. This can lead to many gaps in the resulting visualization in each layer. Formally, a gap in layer L_i is a maximal consecutive sequence of dummy nodes (Figure 1c). We argue that this hinders readability; thus, we extend algorithms for OSCM to only allow (1) *side gaps*, that is, one gap on the left and one gap on the right of a layer, or (2) at most k *gaps* for each layer. Gaps have already been motivated by Nöllenburg and Wallinger for hive plots [10], which is essentially a circular variant of the Sugiyama framework with some additional features.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.

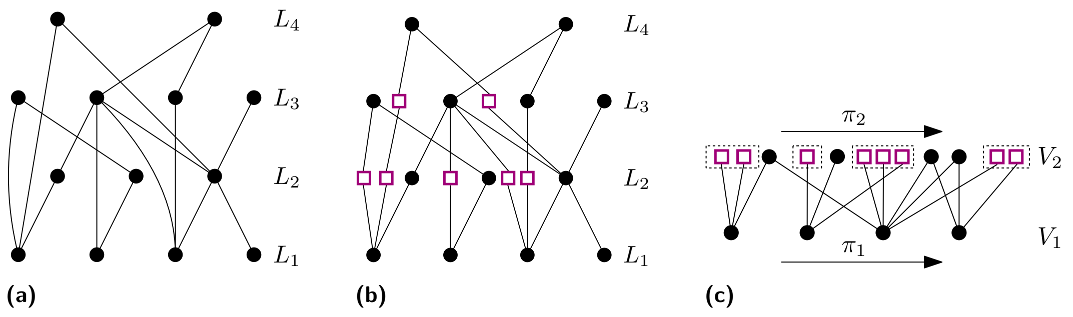


Figure 1 (a) A layered graph drawing. (b) Long edges are replaced by paths of dummy nodes, shown as violet squares. (c) A drawing of two layers with the two node orderings π_1 and π_2 such that π_2 has 4 gaps (shown with dashed rectangles), two of which are side gaps.

Nöllenburg and Wallinger have introduced gaps at fixed positions of each layer, including the variant of side gaps. We extend their work by introducing the variant of k gaps, where the gaps can be placed arbitrarily. Furthermore, we consider the problem from a more theoretical perspective, proving approximation ratios of our algorithms. For both variants, side gaps and k gaps, we propose approximations and exact algorithms that are experimentally evaluated.

Related work. The well-known Sugiyama framework [12] for layered graph drawing serves as the main motivation of this work. As mentioned, a key step of this framework is to minimize crossings between two adjacent layers by permuting the order of nodes of one layer while keeping the second layer fixed, which is a known NP-hard problem called one-sided crossing minimization (OSCM) [4]. There exist heuristics with approximation guarantees [4, 12, 9], FPT-algorithms parameterized by the number of crossings [3, 8], and exact algorithms based on integer linear programs [6].

A restricted variant of OSCM has already been studied by Forster [5], where the relative order of node pairs can be restricted; thus the computed order has to conform to a given partial order. This is different to restrictions on gaps, which cannot be represented by partial orders. Further, Nöllenburg and Wallinger [10] have considered gaps in a circular drawing style of graphs, called *hive plots*. Our theoretical results are of independent interest to their work, and we extend their setting of gaps at fixed positions to gaps at arbitrary positions.

Gaps can also be regarded as groups of edges that can be bundled together. Edge bundling has already been applied in the context of layered graph drawing [11].

Structure. We state the formal problems for our OSCM-variants in Section 2. In Section 3 and Section 4 we give polynomial time approximation algorithms for the respective problems. In the full version of this paper [1] we give complete proofs for statements marked with $\star$, exact ILP formulations, and an experimental evaluation of selected algorithms. The source code is available online [2].

2 Preliminaries

Permutations. We treat permutations π as lists of a set X. Two permutations π, π' of disjoint sets can be concatenated by $\pi \star \pi'$. For $x, y \in \pi$ we write $x \prec_\pi y$ if x comes before y in π. For $X' \subseteq X$, $\pi[X']$ is the *induced* permutation on X', i.e., for all $x, y \in X'$, $x \prec_\pi y$ iff. $x \prec_{\pi[X']} y$. Further, $\Pi(X)$ denotes the set of all permutations of X, and $\pi[i : j]$ is the set of elements in π whose index is between i and j inclusively, using 1-indexing.

One-sided crossing minimization. The problems discussed in this paper have as input a bipartite graph $G = (V_1 \dot{\cup} V_2, E)$, $E \subset V_1 \times V_2$. We set $n := |V_1 \cup V_2|$ and $m := |E|$. The classic one-sided crossing minimization (OSCM) problem is given G and a permutation π_1 of V_1. The task is to find a permutation π_2 of V_2 that minimizes the number of edge crossings in a two-layer straight-line drawing of G such that nodes in V_1 are ordered according to π_1 on the bottom layer and nodes in V_2 are ordered according to π_2 on the top layer. Such crossings can be determined combinatorially by π_1 and π_2; namely, edges $e = (u_1, u_2)$ and $e = (v_1, v_2)$, $u_i, v_i \in V_i$, cross w.r.t. π_1 and π_2 if $u_1 \prec_{\pi_1} v_1$ and $u_2 \succ_{\pi_2} v_2$, or $u_1 \succ_{\pi_1} v_1$ and $u_2 \prec_{\pi_2} v_2$. Let $\mathrm{cr}(G, \pi_1, \pi_2)$ be the number of crossings determined in such a way. Given G and π_1, OSCM asks for π_2 minimizing $\mathrm{cr}(G, \pi_1, \pi_2)$. Throughout the paper we assume π_1 as fixed. By slightly abusing the notation of cr, we furthermore define for $S, S' \in V_2$, $S \cap S' = \emptyset$ the value $\mathrm{cr}(G, S, S')$ as follows. Let π_2 be any permutation such that all nodes in S come before all nodes in S'. The value $\mathrm{cr}(G, S, S')$ is the number of pairs e, e' that cross w.r.t. π_1 and π_2 such that $e \cap S \neq \emptyset$ and $e' \cap S' \neq \emptyset$.

We extend the OSCM problem by restricting the amount of allowed *gaps*. For this, we note that V_2 consists of the disjoint union of V_2^{r} and V_2^{dm}, where V_2^{r} is the set of *real* nodes and V_2^{dm} is the set of *dummy* nodes obtained by the preprocessing steps performed by the Sugiyama framework [12]. It is important to note that dummy nodes have degree one, which we exploit in all our algorithms. A gap in π_2 is a maximal consecutive sequence of dummy nodes, and $\mathrm{gaps}(\pi_2)$ is the amount of gaps in π_2. Furthermore, a *side gap* is a gap that either contains the leftmost or rightmost dummy nodes in π_2, π_2 is a *side-gap permutation* if all of its gaps are side-gaps. In our restricted OSCM variants, we either allow only side gaps in π_2, or at most k gaps overall. The formal problems are given below, starting with side gaps.

▶ **Problem 1** (OSCM-SG). Given a bipartite graph $G = (V_1 \dot{\cup} V_2, E)$ and a permutation π_1 of V_1, find a permutation $\pi_2 \in \Pi(V_2)$ such that π_2 is a side-gap permutation and $\mathrm{cr}(G_1, \pi_1, \pi_2)$ is minimal.

▶ **Problem 2** (OSCM-kG). Given a bipartite graph $G = (V_1 \dot{\cup} V_2, E)$, a permutation π_1 of V_1, and $k \in \mathbb{N}$, find a permutation $\pi_2 \in \Pi(V_2)$ such that $\mathrm{gaps}(\pi_2) \leq k$ and $\mathrm{cr}(G_1, \pi_1, \pi_2)$ is minimal.

Clearly, both problems are NP-hard as they are equivalent to classic OSCM which is NP-hard [4], if we have no dummy nodes.

3 Approximation Algorithms for OSCM-SG

We show that any approximation algorithm for the classic OSCM problem can be transformed to an approximation algorithm for OSCM-SG with the same approximation ratio. First, we show in the below lemma that there will never be edge crossings that involve two edges that are both incident to a dummy node in an optimal solution to OSCM-SG and OSCM-kG.

▶ **Lemma 3.1** ($\star$). *Given π_1, π_2 such that a pair of edges $e, e' \in V_1 \times V_2^{dm}$ crosses, there is π_2' such that (1) $cr(G, \pi_1, \pi_2') < cr(G, \pi_1, \pi_2)$, (2) $gaps(\pi_2') \leq gaps(\pi_2)$, and (3) if π_2 is a side-gap permutation, so is π_2'.*

Due to the above lemma, we fix in the rest of the paper π_2^{dm} as the order of V_2^{dm} determined by sorting V_2^{dm} ascending by their neighbor's position in π_1. Dummy nodes with the same neighbor can be ordered arbitrarily. If for any solution π_2, $\pi_2[V_2^{\mathrm{dm}}] \neq \pi_2^{\mathrm{dm}}$, we can transform π_2 into π_2' having properties (1)-(3) of Lemma 3.1 and with $\pi_2'[V_2^{\mathrm{dm}}] = \pi_2^{\mathrm{dm}}$.

Now, given an algorithm $\mathcal{A}$ for OSCM with approximation ratio α, we get an approximation algorithm for OSCM-SG with the same approximation ratio, as given below.

▶ **Theorem 3.2** (⋆). *Let $\mathcal{A}$ be an algorithm for OSCM with approximation ratio α and runtime $\mathcal{O}(f(n,m))$. Then there exists an algorithm $\mathcal{B}$ for OSCM-SG with approximation ratio α and runtime $\mathcal{O}(f(n,m)+m)$.*

The key idea for the algorithm $\mathcal{B}$ is that we can in polynomial time compute the optimal placement of gap nodes and this placement is independent of the ordering of real nodes. If $\mathcal{A}$ is for example the median heuristic [4], then Theorem 3.2 gives us a polynomial time 3-approximation algorithm for OSCM-SG. For an exact algorithm, we can substitute for $\mathcal{A}$ any exact algorithm for OSCM ($\alpha = 1$) such as ILP formulations [6].

4 Approximation Algorithms for OSCM-kG

Adapting heuristics for OSCM-kG is not as straight-forward. This is because once we have determined $\pi_2[V_2^{\mathrm{r}}]$, we have to consider all possibilities of inserting dummy nodes without having more than k gaps. Furthermore, now the optimal placement of dummy nodes is dependent on $\pi_2[V_2^{\mathrm{r}}]$. We will only be able to extend OSCM heuristics with the following property.

▶ **Definition 4.1.** Let $\mathcal{A}$ be an algorithm for OSCM, (G,π_1) be any instance of OSCM with $G=(V_1\dot\cup V_2,E)$. Consider a set of new nodes V', $E'\subseteq V_1\times V'$, and $G'=(V_1\dot\cup(V_2\cup V'),E\cup E')$. Now apply $\mathcal{A}$ to (G,π_1) and to (G',π_1) to obtain solutions π_2 and π_2', respectively. The algorithm $\mathcal{A}$ is *dummy-independent* if $\pi_2[V_2]=\pi_2'[V_2]$ always holds.

Examples of dummy-independent algorithms are for example the barycenter-, and median-heuristic. By plugging $V'=V_2^{\mathrm{dm}}$ in the above definition, we see that the order of real nodes computed by $\mathcal{A}$ is independent of the dummy nodes in G, when $\mathcal{A}$ is dummy-independent.

We can now extend any dummy-independent approximation algorithm $\mathcal{A}$ to OSCM-kG maintaining the approximation ratio.

▶ **Theorem 4.2.** *Let $\mathcal{A}$ be a dummy-independent algorithm for OSCM with approximation ratio α and runtime $\mathcal{O}(f(n,m))$. Then there exists an algorithm $\mathcal{B}$ for OSCM-kG with approximation ratio α and runtime $\mathcal{O}(f(n,m)+|V_2^{r}|\cdot|V_2^{dm}|^2\cdot k)$.*

Proof. The algorithm $\mathcal{B}$ first determines $\pi_2^{\mathrm{r}} := \pi_2[V_2^{\mathrm{r}}]$ by applying $\mathcal{A}$ to the OSCM instance $(G[V_1\cup V_2^{\mathrm{r}}],\pi_1)$. We define a dynamic program to merge the two orders π_2^{r} and π_2^{dm}. The dynamic programming table DP contains entries $DP[g,i,j]$ which represents the minimum number of crossings between edge pairs e,e', e being incident to V_2^{r} and e' being incident to V_2^{dm}, using at most g gaps when merging the first i nodes in π_2^{r} and the first j nodes in π_2^{dm}; further, g goes from 0 to k, i goes from 0 to $|V_2^{\mathrm{r}}|$, and j goes from 0 to $|V_2^{\mathrm{dm}}|$. The base cases are

- $DP[0,i,0]=0$ for $0\le i\le |V_2^{\mathrm{r}}|$,
- $DP[0,i,j]=\infty$ for $0\le i\le |V_2^{\mathrm{r}}|$, $1\le j\le |V_2^{\mathrm{dm}}|$,

and the transitions for $g>0$ can be computed as

$$\begin{aligned}DP[g,i,j] = \min_{0\le j'\le j}\,&[DP[g-1,i,j']\\&+\mathrm{cr}(G,\pi_2^{\mathrm{r}}[1:i],\pi_2^{\mathrm{dm}}[j'+1:j])+\mathrm{cr}(G,\pi_2^{\mathrm{dm}}[j'+1:j],\pi_2^{\mathrm{r}}[i+1:|V_2^{\mathrm{r}}|])].\end{aligned}$$

The optimal number of crossings can be read from $DP[k,|V_2^{\mathrm{r}}|,|V_2^{\mathrm{dm}}|]$, and the corresponding permutation can be reconstructed from the entries in DP. The runtime can be achieved by precomputing $\mathrm{cr}(G,\pi_2^{\mathrm{r}}[1:i],\pi_2^{\mathrm{dm}}[j'+1:j])$ and $\mathrm{cr}(G,\pi_2^{\mathrm{dm}}[j'+1:j],\pi_2^{\mathrm{r}}[i+1:|V_2^{\mathrm{r}}|])$.

For correctness, consider an optimal solution π_2^{opt} with c crossings. By Lemma 3.1 no edge pairs incident to V_2^{dm} cross. Now contract each set of dummy nodes that appear in a gap together, obtaining the graph G'. Apply $\mathcal{A}$ to (G', π_1), obtaining a solution π_2' with at most $\alpha \cdot c$ crossings. Now revert the contraction and replace each contracted node by its original sequence of dummy nodes in π_2'. The newly obtained permutation is in the solution space of the dynamic program because $\pi_2^{\text{r}} = \pi_2'[V_2^{\text{r}}]$ ($\mathcal{A}$ is dummy-independent), hence we have an α-approximation. ◀

5 Summary of the Experimental Evaluation and Conclusion

In an experimental evaluation (see full version [1]) we compared exact ILP formulations, and heuristics for OSCM-SG and OSCM-kG. In particular, we applied the algorithms of Theorem 3.2 and Theorem 4.2 to the barycenter and median heuristics. Our evaluation showed that both heuristics perform well when compared with the exact algorithms, with optimality gaps below five percent. This is in line with the performance of the original median and barycenter heuristic when compared with exact algorithms for OSCM [7]. Further, the exact ILP formulations could solve all instances with up to 70 nodes per layer in under ten seconds. We also investigated the impact of the number of gaps k on the number of crossings, showing diminishing returns for increasing k. Namely, the number of crossings decreases significantly from $k = 1$ to $k = 2$, but only slightly for larger k.

Further research is required to properly integrate our algorithms into the Sugiyama framework. In particular, adjustments might be required to guarantee few edge crossings over all layers, not just between a pair of layers. One might also investigate larger instances, i.e., with more vertices or higher edge densities. Additionally, case studies could show how few gaps can reduce the amount of clutter in the layered graph drawing.

References

1 Alexander Dobler and Jakob Roithinger. Layered graph drawing with few gaps and few crossings, 2025. `arXiv:2502.20896`.

2 Alexander Dobler and Jakob Roithinger. Layered graph drawing with few gaps and few crossings, 2025. `doi:10.17605/OSF.IO/ZP58M`.

3 Vida Dujmovic and Sue Whitesides. An efficient fixed parameter tractable algorithm for 1-sided crossing minimization. *Algorithmica*, 40(1):15–31, 2004. `doi:10.1007/S00453-004-1093-2`.

4 Peter Eades and Nicholas C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994. `doi:10.1007/BF01187020`.

5 Michael Forster. A fast and simple heuristic for constrained two-level crossing reduction. In János Pach, editor, *Proc. Symposium on Graph Drawing and Network Visualization (GD'04)*, volume 3383 of *Lecture Notes in Computer Science*, pages 206–216. Springer, 2004. `doi:10.1007/978-3-540-31843-9\_22`.

6 Michael Jünger and Petra Mutzel. Exact and heuristic algorithms for 2-layer straightline crossing minimization. In Franz-Josef Brandenburg, editor, *Proc. Symposium on Graph Drawing and Network Visualizations (GD'95)*, volume 1027 of *Lecture Notes in Computer Science*, pages 337–348. Springer, 1995. `doi:10.1007/BFB0021817`.

7 Michael Jünger and Petra Mutzel. 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. *J. Graph Algorithms Appl.*, 1(1):1–25, 1997. `doi:10.7155/JGAA.00001`.

8 Yasuaki Kobayashi and Hisao Tamaki. A faster fixed parameter algorithm for two-layer crossing minimization. *Inf. Process. Lett.*, 116(9):547–549, 2016. doi:10.1016/J.IPL.2016.04.012.

9 Hiroshi Nagamochi. An improved approximation to the one-sided bilayer drawing. In Giuseppe Liotta, editor, *Proc. Graph Drawing and Network Visualization (GD'03)*, volume 2912 of *Lecture Notes in Computer Science*, pages 406–418. Springer, 2003. doi:10.1007/978-3-540-24595-7_38.

10 Martin Nöllenburg and Markus Wallinger. Computing hive plots: A combinatorial framework. *J. Graph Algorithms Appl.*, 28(2):101–129, 2024. doi:10.7155/JGAA.V28I2.2990.

11 Sergey Pupyrev, Lev Nachmanson, and Michael Kaufmann. Improving layered graph layouts with edge bundling. In Ulrik Brandes and Sabine Cornelsen, editors, *Proc. Graph Drawing and Network Visualization (GD'10)*, volume 6502 of *Lecture Notes in Computer Science*, pages 329–340. Springer, 2010. doi:10.1007/978-3-642-18469-7_30.

12 Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst. Man Cybern.*, 11(2):109–125, 1981. doi:10.1109/TSMC.1981.4308636.

The Analytic Arc Cover Problem and its Applications to Contiguous Art Gallery, Polygon Separation, and Shape Carving

Eliot W. Robson[1], Jack Spalding-Jamieson[2], and Da Wei Zheng[3]

1 Department of Computer Science, University of Illinois Urbana-Champaign
erobson2@illinois.edu

2 Independent
jacksj@uwaterloo.ca

3 Department of Computer Science, University of Illinois Urbana-Champaign
dwzheng2@illinois.edu

Abstract

We show the following problems are in P:

1. The contiguous art gallery problem – a variation of the art gallery problem where each guard can protect a contiguous interval along the boundary of a simple polygon. This was posed at the open problem session at CCCG '24 by Thomas C. Shermer.
2. The polygon separation problem for line segments – For two sets of line segments S_1 and S_2, find a minimum-vertex convex polygon P that completely contains S_1 and does not contain or cross any segment of S_2.
3. Minimizing the number of half-plane cuts to carve a 3D polytope.

To accomplish this, we study the *analytic arc cover problem* – an interval set cover problem over the unit circle with infinitely many implicitly-defined arcs, given by a function.

Related Version Full Version: [18]

1 Introduction

Many classic problems in computational geometry are minimum covering problems. One class of examples are art gallery problems [15, 21, 24, 16] which asks for the minimum number star-shaped polygons that cover a given polygon. Each star-shaped polygon describes a region that can be seen by a single guard. Some variants of art gallery allow guards to be placed anywhere, while others only allow guards to be placed at a finite set of points (such as vertices of a polygon). Many variants of the former type are $\exists\mathbb{R}$-complete, while variants of the latter type are almost universally in NP, and are often in P. This distinction is not surprising: When there are infinitely many choices for the covering sets, it is often not clear if the problem is even in NP.

In this work, we study three problems with no immediate proofs that they are in any complexity class smaller than $\exists\mathbb{R}$. The first problem is a variant of the art gallery problem with a restriction that each guard can only be responsible for a contiguous region along the boundary. The other two problems relate to separation of geometric objects. We will show that each of these problems can be reduced to a problem we call the *analytic arc cover* problem, which is a set cover problem permitting infinitely many possible sets with some additional structure. We establish machinery for dealing with some "well-behaved" infinite classes of possible covering sets, allowing us to show all three of our problems are in P.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.

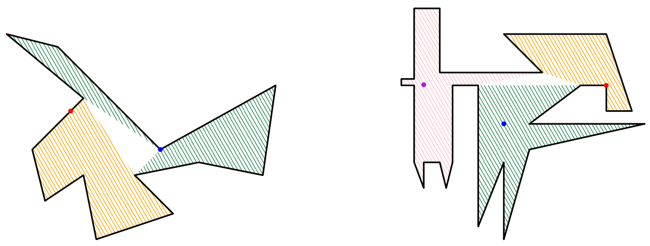


Figure 1 Some examples of CONTIGUOUSARTGALLERY, with optimal solutions. On the right, it is necessary to place a guard at a non-vertex point.

1.1 Art Gallery Problem

Given a simple polygon P and points $x, y \in P$, a guard standing at x sees the point y if the line segment xy is contained in P. A set S of points is said to *guard* the polygon P if every point of P can be seen by some guard. Minimizing the cardinality of S is known as the ARTGALLERY problem[1], and is a very well-studied problem in computational geometry (see one of the numerous surveys dedicated to the problem [15, 21, 24, 16]). This problem was first formulated by Victor Klee in 1976 (see O'Rourke [15]). Many variations have been studied since then, such as when the guards are restricted to the boundary or the vertices of the polygon. There are other variants like the BOUNDARYARTGALLERY problem where guards are allowed to be anywhere but only the boundary of P needs to be guarded.

All of the aforementioned variants were shown to be NP-hard [3, 13, 12]. It can be seen that variants that allow for guards in the interior may not even be in NP, since it was shown that an optimal solution to ARTGALLERY may need guards at irrational points [1]. Further evidence towards this was recently given by Abrahamsen, Adamaszek, and Miltzow [2], who showed that Victor Klee's original formulation is $\exists\mathbb{R}$-complete. Subsequent work by Stade [22] shows that many of the variants of art gallery are also $\exists\mathbb{R}$-complete.

One salient feature of many of the hardness proofs for ARTGALLERY and its variations is that a single guard may be responsible for disjoint regions of the polygon, even with respect to guarding just the boundary of P. In contrast, we can consider the CONTIGUOUSARTGALLERY problem – a variation of the BOUNDARYARTGALLERY problem where each guard can only be responsible for a contiguous section of the boundary (see fig. 1). This problem was described by Thomas C. Shermer at the CCCG '24 open problem session, where he asked the following question:

Is the guarding of disjoint regions necessary for the hardness proofs of ARTGALLERY and variations like BOUNDARYARTGALLERY?
What is the complexity of CONTIGUOUSARTGALLERY? Is it even in NP?

1.2 Minimum Polygon Separation

Given a convex polygon P contained inside another convex polygon Q, the minimum polygon separation problem asks for the polygon S with the minimum number of vertices such that S contains P and is contained within Q. We denote this problem as POLYGONSEPARATION. POLYGONSEPARATION was first studied by Aggarwal, Booth, O'Rourke, Suri, and Yap [4],

[1] Throughout this paper, we will abuse notation and refer to problems like ARTGALLERY as both the optimization variant to minimize the set of guards, as well as the decision variant, where we wish to decide if k guards are sufficient.

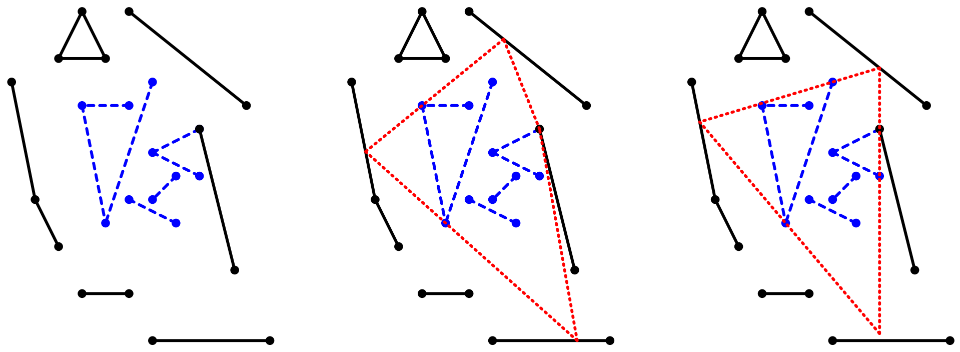


Figure 2 An instance of SEGMENTSEPARATION, and two potential solutions (one sub-optimal).

who also showed that any minimal S is convex. This problem is related to the problem of finding a convex polygon with the minimum number of sides that separates two sets of points S_1 with S_2 that was studied by Edelsbrunner and Preparata [10]. We denote this problem as POINTSEPARATION.

One feature of the algorithm of [4] is that they work in the real RAM model of computation [20], where it is assumed that elementary operations (like $+$, $-$, $\times$, and $\div$) can be performed in $O(1)$ time on real numbers. However, the algorithm they present involves iteratively composing many functions together, and ultimately solving an equation. While [4] claim that their algorithm runs in $O(n \log n)$ time when the input consists of n points, it is unclear what the runtime of the algorithm is in other models of computation (e.g. in a Turing Machine model where inputs are points with rational coordinates). This is because their algorithm may compose $O(n)$ different functions, leading to a function with very high bit complexity, while [4] assumes an equation with such a function can be solved in $O(1)$ time.

One can also consider another similar problem: The problem of separating two collections of line segments by a convex polygon with the minimum number of sides. Feasibility for this problem is easy — one only needs to verify that the convex hull of the "inner" line segments does not contain any part of any of the "outer" line segments in its interior. We call this problem SEGMENTSEPARATION. It can be seen that SEGMENTSEPARATION generalizes the problem of POLYGONSEPARATION when we wish to separate the edges of an outer polygon with the inner polygon. Furthermore, the POINTSEPARATION problem is a special case of SEGMENTSEPARATION when all segments are of zero length. To the best of our knowledge, SEGMENTSEPARATION has not been studied before, and the algorithms of [4] and [10] do not immediately work for this problem.

Moreover, while the problems POLYGONSEPARATION and POINTSEPARATION were implicitly shown to be in P using the algorithms of [4] and [10] respectively (with a larger time complexity in the Turing Machine model of computation rather than the real RAM models used in those papers), the same techniques do not immediately apply to SEGMENTSEPARATION, so it is not immediately clear that SEGMENTSEPARATION is even in NP.

1.3 Polytope Carving

Related to POLYGONSEPARATION is the problem of *carving* one shape out of another. One could also carve a shape out of an arbitrary containing shape, which is equivalent to asking for a set of cuts separating the shape from the ambient space. Two-dimensional carving was first studied by Overmars and Welzl [17], where they aimed to find the cheapest sequence

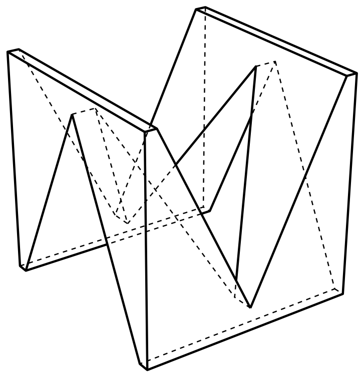


Figure 3 A polytope that can be carved by 10 half-plane cuts.

of line cuts to carve out a convex polygon out of a piece of flat material. It has also been studied in the context of rays cuts [7, 23] and line segments cuts [8, 9].

Three dimensional versions of carving have also been studied in the form of plane cuts [5], line cuts [11], half-plane cuts [19], and "sweeping" ray cuts [19]. The prior work of Robson, Spalding-Jamieson and Zheng [19] on half-plane cuts classified the polytopes that could be carved, but they left open the question of computing the minimum number of half-plane cuts needed to carve out a specified 3D polytope from the ambient space of $\mathbb{R}^3$.

1.4 Our Contribution

We answer the question posed by Thomas C. Shermer with the following theorem, showing that adding contiguity constraints for BoundaryArtGallery makes the problem significantly simpler. We also show that SegmentSeparation is in P.

▶ **Theorem 1.1.** *ContiguousArtGallery is in* P*.*

▶ **Theorem 1.2.** *SegmentSeparation is in* P*.*

We show that a question about minimizing the number of half-plane cuts to carve a 3D polytope, reduces to multiple instances of SegmentSeparation. By the theorem above, we can conclude that finding the minimum cuts to carve a polytope is also in P.

▶ **Theorem 1.3.** *Minimizing the number of half-plane cuts to carve a 3D polytope is in* P*.*

To prove our results, we reduce both SegmentSeparation and ContiguousArtGallery to a problem we call AnalyticArcCover. This problem is a version of the interval cover problem on a circle with infinite number of intervals. Each interval is described by the counterclockwise segment between two points a and b on the circle S^1. We denote this as a half-open interval $[a, b)$, containing a, and not containing b, and call this an **arc**. This infinite set of intervals is given implicitly as a function, as can be seen in the following definition.

▶ **Definition 1.4.** Let $g : S^1 \to S^1$ be a function that maps points on the unit circle S^1 to other points on the unit circle S^1. AnalyticArcCover asks: given g, find the minimum set $X \subset S^1$ such that the set of counter-clockwise arcs $\{[x, g(x)) : x \in X\}$ covers S^1.

We show that ContiguousArtGallery reduces to AnalyticArcCover with piecewise linear-rational functions over a unit-interval representation, and SegmentSeparation reduces to AnalyticArcCover with a two-dimensional analogue. We crucially use the fact that a compositions of two linear rational functions yields another linear rational function, and our two-dimensional analogue also has an analogous property.

1.5 Concurrent work

Two other groups also concurrently investigated the ContiguousArtGallery problem. Surprisingly, both groups devised different high-level approaches to the problem.

Merrild, Rysgaard, Schou, and Svenning [14] give a polynomial time algorithm for the ContiguousArtGallery problem in the real RAM model of computation, but do not bound the bit complexity of the intermediate numbers produced by the algorithm. They posed the question of membership in Pas an open problem, which our results address. They also posed the question of an algorithm for polygons with holes, to which our methods extend (see the end of Section 3 in the full version [18]).

Biniaz, Maheshwari, Mitchell, Odak, Polishchuk, and Shermer [6] also provide a polynomial time algorithm for the ContiguousArtGallery problem. In particular, like us, they were able to provide one running on a Turing machine, implying membership in P. Their approach is more combinatorial than ours, although some small aspects of their proof bear similarities to ours.

In our work, we give a framework for solving a larger class of problems in polynomial time on a Turing machine (i.e., membership in P), including the ContiguousArtGallery problem.

1.6 Organization

We give an outline of our approach in the next section, and we include a full version of this paper in the appendix.

2 Outline of approach

Analytic Arc Cover

We study the AnalyticArcCover problem, which is a variation of the interval cover problem on a circle S^1. The *analytic* part of the problem comes from how there are an infinite number of intervals that are defined implicitly by a function. Formally, we have a function $f : S^1 \to S^1$, such that we can use any interval of the form $[x, f(x)]$. We show that this problem is solvable in polynomial time when f is a piecewise linear rational function with rational coefficients and first-order radical endpoints (that is: of the form $a + b\sqrt{c}$ for integers a, b, c). Importantly, such functions are closed under (finite) max and min operations.

Contiguous Art Gallery

We reduce the contiguous art gallery problem to the analytic arc cover problem for a function f which is piecewise linear rational. For intuition, consider a fixed point x on the boundary of a polygonal art gallery P, and consider a guard that can see x and can see as much of the boundary of P as possible in the counterclockwise direction. We show that the behavior of this point as x varies can be described by a piecewise rational function.

To show this, we start by proving it for an extremely simplified form of the problem. Then, we take progressively less simplified forms of the problem, and show that the function for each can be expressed as the minimum or maximum of a polynomial number of cases of the preceding form.

Polygon Separation and Polytope Carving

Similarly, we can reduce the problem of SegmentSeparation where we wish to separate two sets of line segments to an instance of AnalyticArcCover with a piecewise rational function. The function is in some sense less complex than the one for the contiguous art gallery problem, but we make use of a special case of piecewise linear rational functions over two variables instead of one.

We also show that the problem of determining the minimum number of cuts to cut out a 3D polytope reduces to a polynomial number of instances of SegmentSeparation.

References

1 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. Irrational guards are sometimes needed. In Boris Aronov and Matthew J. Katz, editors, *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, volume 77 of *LIPIcs*, pages 3:1–3:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.SOCG.2017.3.

2 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is ∃ℝ-complete. *J. ACM*, 69(1):4:1–4:70, 2022. doi:10.1145/3486220.

3 Alok Aggarwal. *The art gallery theorem: its variations, applications and algorithmic aspects.* PhD thesis, Johns Hopkins University, 1984.

4 Alok Aggarwal, Heather Booth, Joseph O'Rourke, Subhash Suri, and Chee-Keng Yap. Finding minimal convex nested polygons. *Inf. Comput.*, 83(1):98–110, 1989. doi:10.1016/0890-5401(89)90049-7.

5 Syed Ishtiaque Ahmed, Masud Hasan, and Md. Ariful Islam. Cutting a convex polyhedron out of a sphere. *Graphs Comb.*, 27(3):307–319, 2011. doi:10.1007/s00373-011-1018-1.

6 Ahmad Biniaz, Anil Maheshwari, Joseph S. B. Mitchell, Saeed Odak, Valentin Polishchuk, and Thomas Shermer. Contiguous boundary guarding, 2024. URL: https://arxiv.org/abs/2412.15053, arXiv:2412.15053.

7 Ovidiu Daescu and Jun Luo. Cutting out polygons with lines and rays. *Int. J. Comput. Geom. Appl.*, 16(2-3):227–248, 2006. doi:10.1142/S0218195906002014.

8 Erik D. Demaine, Martin L. Demaine, and Craig S. Kaplan. Polygons cuttable by a circular saw. *Comput. Geom.*, 20(1-2):69–84, 2001. doi:10.1016/S0925-7721(01)00036-0.

9 Adrian Dumitrescu and Masud Hasan. Cutting out polygons with a circular saw. *Int. J. Comput. Geom. Appl.*, 23(2):127–140, 2013. doi:10.1142/S0218195913600030.

10 Herbert Edelsbrunner and Franco P. Preparata. Minimum polygonal separation. *Inf. Comput.*, 77(3):218–232, 1988. doi:10.1016/0890-5401(88)90049-1.

11 Jerzy W. Jaromczyk and Mirosław Kowaluk. Sets of lines and cutting out polyhedral objects. *Comput. Geom.*, 25(1-2):67–95, 2003. doi:10.1016/S0925-7721(02)00131-1.

12 Aldo Laurentini. Guarding the walls of an art gallery. *Vis. Comput.*, 15(6):265–278, 1999. doi:10.1007/S003710050177.

13 D. T. Lee and Arthur K. Lin. Computational complexity of art gallery problems. *IEEE Trans. Inf. Theory*, 32(2):276–282, 1986. doi:10.1109/TIT.1986.1057165.

14 Magnus Christian Ring Merrild, Casper Moldrup Rysgaard, Jens Kristian Refsgaard Schou, and Rolf Svenning. The contiguous art gallery problem is solvable in polynomial time, 2024. URL: https://arxiv.org/abs/2412.13938, arXiv:2412.13938.

15 Joseph O'Rourke. *Art gallery theorems and algorithms.* Oxford University Press, Inc., 1987.

16 Joseph O'Rourke. Visibility. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry, Second Edition*, pages 643–663. Chapman and Hall/CRC, 2004. `doi:10.1201/9781420035315.CH28`.

17 Mark H. Overmars and Emo Welzl. The complexity of cutting paper (extended abstract). In *Proceedings of the First Annual Symposium on Computational Geometry, Baltimore, Maryland, USA, June 5-7, 1985*, pages 316–321. ACM, 1985. `doi:10.1145/323233.323274`.

18 Eliot W. Robson, Jack Spalding-Jamieson, and Da Wei Zheng. The analytic arc cover problem and its applications to contiguous art gallery, polygon separation, and shape carving, 2024. URL: `https://arxiv.org/abs/2412.15567`, `arXiv:2412.15567`.

19 Eliot W. Robson, Jack Spalding-Jamieson, and Da Wei Zheng. Carving polytopes with saws in 3d. In Rahnuma Islam Nisha, editor, *Proceedings of the 36th Canadian Conference on Computational Geometry, CCCG 2024, Brock University, St. Catharines, Ontario, Canada, July 17 - August 19, 2024*, pages 145–152, 2024.

20 Michael Ian Shamos. *Computational Geometry.* PhD thesis, Yale University, 1978. URL: `http://euro.ecom.cmu.edu/people/faculty/mshamos/1978ShamosThesis.pdf`.

21 Thomas C. Shermer. Recent results in art galleries (geometry). *Proc. IEEE*, 80(9):1384–1399, 1992. `doi:10.1109/5.163407`.

22 Jack Stade. Complexity of the boundary-guarding art gallery problem. *CoRR*, abs/2210.12817, 2022. `arXiv:2210.12817`, `doi:10.48550/ARXIV.2210.12817`.

23 Xuehou Tan. Approximation algorithms for cutting out polygons with lines and rays. In *Computing and Combinatorics, 11th Annual International Conference, COCOON 2005, Kunming, China, August 16-29, 2005, Proceedings*, volume 3595 of *Lecture Notes in Computer Science*, pages 534–543. Springer, 2005. `doi:10.1007/11533719\_54`.

24 Jorge Urrutia. Art gallery and illumination problems. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 973–1027. North Holland / Elsevier, 2000. `doi:10.1016/B978-044482537-7/50023-1`.

Efficient Shape Reconfiguration by Hybrid Programmable Matter[*]

Jonas Friemel[1], David Liedtke[2], and Christian Scheffer[1]

1 **Bochum University of Applied Sciences, Germany**
`{jonas.friemel, christian.scheffer}@hs-bochum.de`

2 **Paderborn University, Germany**
`liedtke@mail.upb.de`

Abstract

Shape formation is one of the most thoroughly studied problems in most algorithmic models of programmable matter. However, few existing shape formation algorithms utilize similarities between an initial configuration and a desired target shape. In the hybrid model, an active agent with the computational capabilities of a deterministic finite automaton can form shapes by lifting and placing passive tiles on the triangular lattice. We study the shape reconfiguration problem where the agent needs to move all tiles in an input shape to so-called target nodes, which are distinguishable from other nodes. We present a worst-case optimal $\mathcal{O}(mn)$ algorithm for simply connected target shapes where m is the initial number of unoccupied target nodes and n is the size of the target shape.

Related Version arXiv:2501.08663

1 Introduction

In the field of programmable matter, small (possibly nano-scale) particles are envisioned to solve tasks like self-assembling into desired shapes, making coordinated movements, or coating small objects [20]. The particles may be controlled by external stimuli or act on their own with limited computational capabilities. In the future, programmable matter could become relevant for targeted medical treatments [1] or in the form of self-assembling structures in environments that are not easily accessible by humans such as in space [13]. There are multiple computational models of programmable matter that differ in the types of particles, their capabilities, and the underlying graph populated by the particles. The particles in each model generally fall into one of the following two categories: *active* agents that can perform (typically limited) computations and move on the underlying graph by themselves [4, 5, 22], and *passive* entities that do not move or act without external influence [19, 21]. The *hybrid model* combines both aspects [8, 9, 14, 18]. Here, passive hexagonal tiles and an active agent with the computational power of a finite automaton populate the triangular lattice.

A central research problem in programmable matter is shape formation [3, 6, 15, 16, 17]. In the hybrid model, the agent needs to rearrange tiles from an arbitrary initial configuration into a predefined shape such as a line or a parallelogram [9, 10, 11]. Once a shape is formed, it is typically assumed that the agent is finished and the shape will remain intact. Thus, existing shape formation algorithms are not designed to repair small shape defects such as individually misplaced tiles. For example, if an agent executes the triangle construction algorithm presented by Gmyr et al. [9] on a set of tiles that already closely resembles a triangle, it deconstructs the entire structure and rebuilds the shape from scratch.

[*] This work was supported by the DFG projects SCHE 1931/4-1 and SCHE 1592/10-1.

Our contributions. In this extended abstract, we present an algorithm that can take advantage of similarities between an initial and a target shape. A problem instance for the shape reconfiguration problem is given by a set of target nodes, which are distinguishable from non-target nodes by the agent. The algorithm solves the problem for simply connected target shapes with asymptotically worst-case optimal runtime, yielding a speedup over existing shape formation algorithms on instances where the initial and target shape largely overlap. Due to space constraints, many details are deferred to the full version of this paper, where we additionally present an algorithm for a large class of target shapes that may contain holes [7].

2 Preliminaries

In the hybrid model, the triangular lattice $G_\triangle = (V_\triangle, E_\triangle)$ is occupied by a single active agent r with limited computational capabilities and a finite number of passive tiles. We call a node $v \in V$ *tiled* if it is occupied by a tile and denote the set of tiled nodes with T. At any time, a node may be occupied by at most one tile and each tile may only occupy a single node. Similarly, the agent may only occupy a single (tiled or untiled) node at a time. It cannot distinguish any two tiles from one another and it can carry up to one tile. A tuple $C = (T, p)$, where $p \in V_\triangle$ is the node occupied by r, is called a *configuration*. A configuration $C = (T, p)$ has *size* $|T|$. All initial and target configurations in this work have size n. We call a configuration C *connected* if $G_\triangle|_T$ is connected or $G_\triangle|_{T\cup\{p\}}$ is connected and r is carrying a tile. Similarly, C is *simply connected* if $G_\triangle|_T$ is simply connected, i.e., $G_\triangle|_{V_\triangle \setminus T}$ is connected, or $G_\triangle|_{T\cup\{p\}}$ is simply connected and r is carrying a tile.

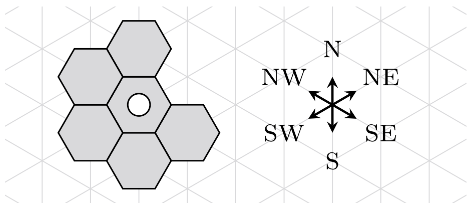


Figure 1 An agent on tiles and the compass directions on the triangular lattice.

The agent has an internal compass to differentiate between the six edge directions on the graph $G_\triangle$ (N, NE, SE, S, SW, NW). For ease of presentation, we assume that this compass aligns with the global directions on the triangular lattice shown in Figure 1. We denote the set of compass directions by $\mathcal{D} := \{\text{N}, \text{NE}, \text{SE}, \text{S}, \text{SW}, \text{NW}\}$ and define $\mathcal{D}$ to be isomorphic to the ring of integers modulo six $\mathbb{Z}/6\mathbb{Z}$ with $\text{N} \equiv 0$, $\text{NE} \equiv 1$, and so on, up to $\text{NW} \equiv 5$. With a slight abuse of notation, this allows us to perform simple additions on directions, e.g., $\text{NE} + 2 = \text{S}$. Intuitively, by adding $\ell \in \mathbb{Z}$ to a direction $d \in \mathcal{D}$, you obtain the next direction from d after ℓ clockwise turns of 60 degrees around the compass shown in Figure 1. For a node $v \in V_\triangle$ and a direction $d \in \mathcal{D}$, the node adjacent to v in direction d on $G_\triangle$ is called a *neighbor of* v (in direction d). We denote the set of all neighbors of v by $N(v)$.

The agent acts in *look-compute-move* cycles. Its "vision" in the *look* phase is limited to neighboring nodes, i.e., it can only see tiles within a hop-distance of ≤ 1 to p. In the *compute* phase, r uses the gathered information to determine its next internal state and its action on the graph. The agent has the computational capabilities of a finite-state automaton. Consequently, it has only constant memory and cannot store a map of the configuration. Finally, r enters the *move* phase, where it may perform any of the following actions: (i) Move to an adjacent (tiled or untiled) node, (ii) lift the tile at p if r is not carrying a tile and connectivity is maintained, and (iii) place a tile at p if r is carrying a tile and p is untiled.

Problem statement. Consider two connected sets of nodes $\mathcal{I}, \mathcal{T} \subseteq V_\triangle$ with $|\mathcal{I}| = |\mathcal{T}| = n$ and a non-empty and connected intersection $\mathcal{I} \cap \mathcal{T}$, and an initial position $p^0 \in \mathcal{I}$ for the agent r. We refer to $\mathcal{I}$ and $\mathcal{T}$ as the *input* and *target shape*, respectively, with the corresponding nodes being referred to as *input* and *target nodes*. An algorithm solves the Shape Reconfiguration Problem, if its execution results in a sequence of connected configurations $C^0 = (T^0, p^0), \ldots, C^t = (T^t, p^t)$ for some $p^t \in V_\triangle$ with $T^0 = \mathcal{I}$ and $T^t = \mathcal{T}$ such that each configuration C^ℓ results from configuration $C^{\ell-1}$ by applying the agent's legal move actions (i)–(iii) to $p^{\ell-1}$ for $0 < \ell \leq t$. In the remainder of this extended abstract, we drop the superscripts for ease of presentation. A node $v \in T \setminus \mathcal{T}$ is called a *supply node* and a node $w \in \mathcal{T} \setminus T$ is called a *demand node*. Throughout this work, the initial number of supply nodes is denoted by $m := |\mathcal{I} \setminus \mathcal{T}| = |\mathcal{T} \setminus \mathcal{I}|$. Finally, tiles on target nodes are called *target tiles* and tiles on supply nodes are called *supply tiles*. Thus, to solve the Shape Reconfiguration Problem, an agent needs to move all m supply tiles to the m demand nodes, see Figure 2.

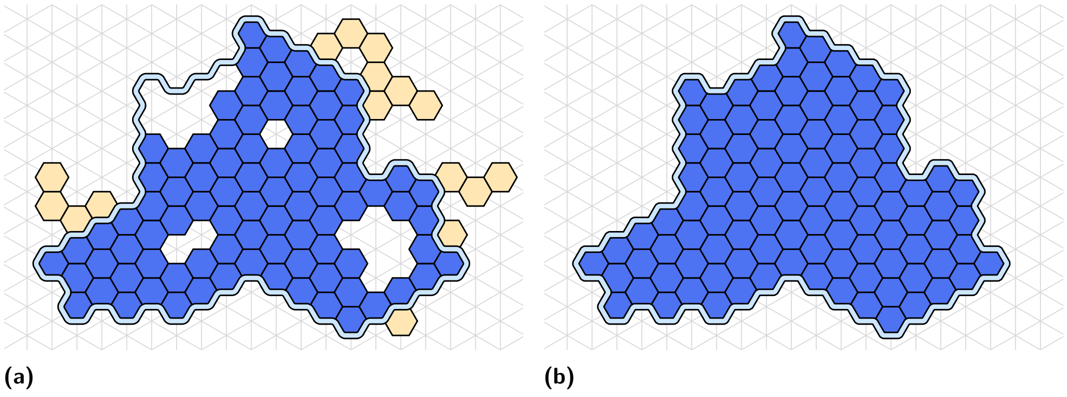


Figure 2 An example instance of the Shape Reconfiguration Problem. The light blue line encircles the target shape $\mathcal{T}$. The blue tiles are target tiles. The yellow tiles are supply tiles and need to be moved to demand nodes. **(a)** shows the tiles in the input shape $\mathcal{I}$ and **(b)** shows the final shape after all supply tiles have been moved to the target shape $\mathcal{T}$.

When r is in the *look* phase of a look-compute-move cycle, it can determine which of the nodes within its visibility range are target nodes. This assumption does not make our agent more powerful than an agent r' that can only query $\mathcal{T}$ for its own position p' as r' could simply visit all six adjacent nodes within a constant number of steps to get the same information. Some adjacent nodes may be unreachable without violating the connectivity constraint, but our agent ignores these nodes in the algorithm presented here.

Boundaries and holes. Let $S \subseteq V_\triangle$ be a finite subset of nodes and let M be the unique infinite node set among the node sets of all connected components of $G_\triangle|_{V_\triangle \setminus S}$. All finite connected components of $G_\triangle|_{V_\triangle \setminus S}$ are called *holes* of S. If S is simply connected, it has no holes. The set $B(S) := \bigcup_{v \in M} S \cap N(v)$ is called the *boundary* and M is called the *outside* of S. For any node $w \in S$, let S_w be the node set of the connected component of $G_\triangle|_S$ containing w and let M_w be the unique infinite node set among the node sets of all connected components of $G_\triangle|_{V_\triangle \setminus S_w}$. Then $B_w(S) := \bigcup_{v \in M_w} S \cap N(v)$ is called the w-boundary of S. We refer to $B(\mathcal{T})$ as the *target boundary*, to $B(T \cap \mathcal{T})$ as the *target tile boundary*, and to $B_w(T \setminus \mathcal{T})$ as the *boundary of a supply component* for a supply node $w \in T \setminus \mathcal{T}$, see Figure 3.

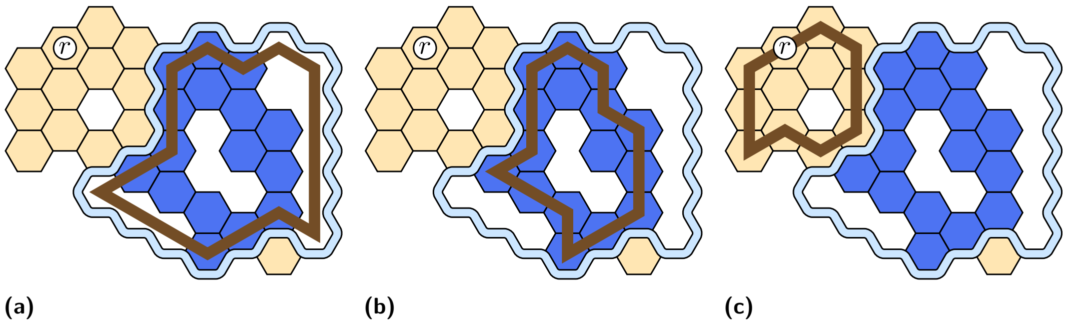


Figure 3 Boundaries **(a)** $B(\mathcal{T})$, **(b)** $B(T \cap \mathcal{T})$, and **(c)** $B_p(T \setminus \mathcal{T})$. The agent r is on node p.

3 Simply Connected Shape Reconfiguration

We present a worst-case optimal algorithm to solve the Shape Reconfiguration Problem for simply connected target shapes. Note that the input shape may contain holes. The algorithm we present here is non-terminating, i.e., the agent does not stop its execution even after the target shape is already built. In the full version of this paper, we also present a terminating algorithm that works on a larger class of target shapes [7].

The algorithm is subdivided into four phases which are executed one after another. The last three phases are repeated until the target shape is formed.

- FindBoundary: The agent r traverses T until it enters the target tile boundary $B(T \cap \mathcal{T})$.
- FindSupply: Since the target shape $\mathcal{T}$ is simply connected, every connected component of supply tiles is adjacent to this boundary. Thus, to reach a supply tile, r merely needs to traverse $B(T \cap \mathcal{T})$ by the well-known *left-hand rule* (LHR), see Figure 4a.
- CompactSupply: Once r reaches a supply tile, it traverses the corresponding supply component until it finds a safely removable tile, i.e., a that can be lifted without breaking connectivity. This requires the agent to first reconfigure the component itself since a finite automaton cannot always find safely removable tiles on tile shapes with holes whereas finding tiles that can be moved to adjacent nodes without violating connectivity is possible [9]. The agent strategically compacts the supply component by moving supply tiles away from the outside of the component's boundary $B_p(T \setminus \mathcal{T})$ whenever possible. This way, it "creates" safely removable tiles. Once r reaches such a tile, it lifts it and returns to $B(T \cap \mathcal{T})$. An example execution of this phase is given in Figure 5.
- FindDemand: Traversing the boundary of the target shape is not sufficient as some components of demand nodes may be fully enclosed by tiled target nodes. To find them, the agent traverses all *columns* of the target shape, which are maximal connected lines of target nodes. Since $\mathcal{T}$ is simply connected, all columns have nodes on the target boundary $B(\mathcal{T})$. Thus, the agent can simply traverse $B(\mathcal{T})$ by the LHR and traverse a column after each step until it eventually finds a demand node where it can place its carried tile, see Figure 4b.

In phase FindBoundary, we make use of an existing shape formation algorithm where all tiled nodes are eventually visited. The agent stops as soon as it finds a tiled target node with a non-target neighbor to initialize a pointer to the outside of $B(T \cap \mathcal{T})$. In the full version [7], we show that r reaches the target tile boundary within $\mathcal{O}(mn)$ time steps by executing the block formation algorithm presented by Gmyr et al. [9].

▶ **Lemma 3.1.** *An agent can find the target tile boundary $B(T \cap \mathcal{T})$ in $\mathcal{O}(mn)$ time steps on instances of the Shape Reconfiguration Problem with simply connected target shapes.*

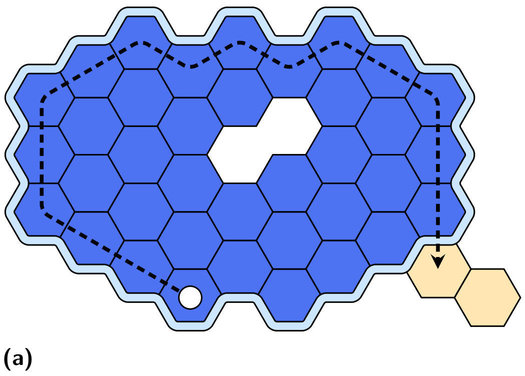


Figure 4 The traversal paths of an agent in phases **(a)** FindSupply and **(b)** FindDemand.

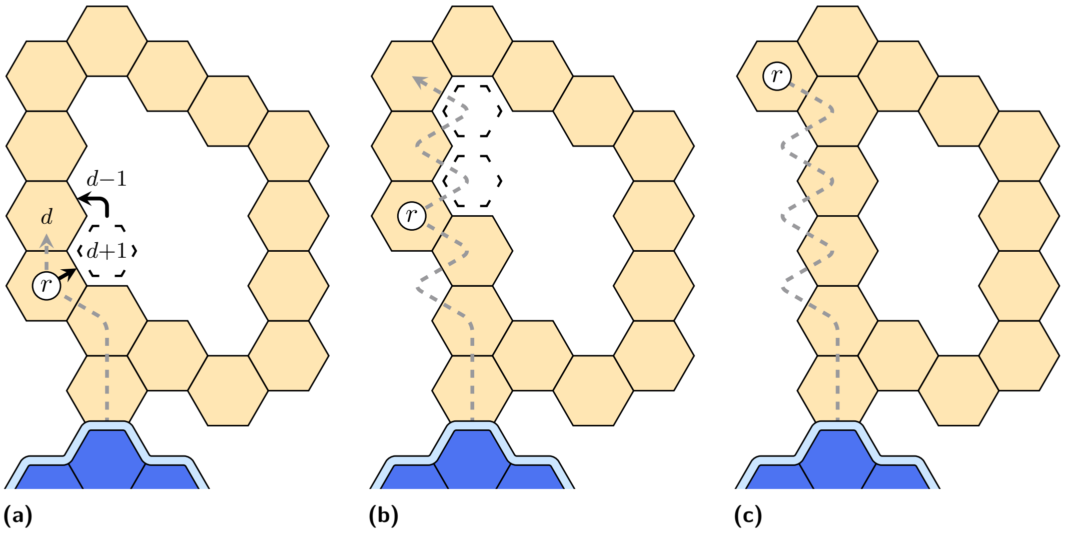


Figure 5 Phase CompactSupply. The agent r traverses $B_p(T \setminus \mathcal{T})$ (gray line) until it enters a supply tile in **(a)** that can be moved inward. Direction $d = \text{N}$ is the next LHR movement direction, but the highlighted node in direction $d+1 = \text{NE}$ is untiled, so r moves its current tile there and then continues its traversal by moving in direction $d-1 = \text{NW}$. In **(b)**, the agent continues the same process for the following two tiles. Finally, r ends on a safely removable tile in **(c)**.

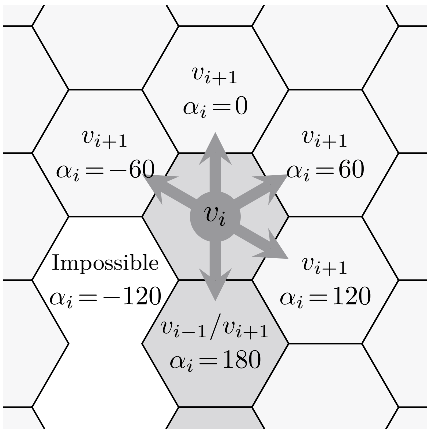

v_{i+1} $\alpha_i = 0$
v_{i+1} $\alpha_i = -60$
v_{i+1} $\alpha_i = 60$
v_i
Impossible $\alpha_i = -120$
v_{i+1} $\alpha_i = 120$
v_{i-1}/v_{i+1} $\alpha_i = 180$

Figure 6 Path turn degrees α_i and successor nodes v_{i+1} in the proof sketch of Lemma 3.2.

The agent then spends at most $\mathcal{O}(n)$ time steps in the remaining phases between each pickup and placement of a tile. We focus on the technically more challenging phase CompactSupply. For details, we refer to the full version [7].

▶ **Lemma 3.2.** *If the agent is in phase CompactSupply, it switches to phase FindDemand carrying a tile within $\mathcal{O}(n)$ time steps.*

Proof Sketch. It suffices to show that r encounters a safely removable supply tile during phase CompactSupply. If the supply component is simply connected, this happens within an LHR traversal of $B_p(T \setminus \mathcal{T})$. We thus focus on the case where the component is not simply connected and no safely removable supply tile is encountered during the traversal.

The agent r traverses $B_p(T \setminus \mathcal{T})$ by the LHR. Internally, it stores a pointer d to the next node in an LHR traversal which is updated at each step. If the nodes in directions $d-3$, $d-2$, $d-1$, and $d+1$ are untiled (or in $\mathcal{T}$), r moves the tile at p in direction $d+1$, see Figure 5a.

Let $P = (v_1, \dots, v_k)$ with $v_1 = p$ and $k = \mathcal{O}(n)$ be the path of a full LHR traversal around $B_p(T \setminus \mathcal{T})$, i.e., the traversal repeats after v_k. We can show that r encounters a safely removable tile while traversing P and moving tiles as specified above. To do so, let $\alpha_i \in (-180, 180]$ be the degree by which r needs to turn to the right to move from v_i to v_{i+1}, see Figure 6. Note that the tile at v_i is safely removable if $|\alpha_i| > 60$.

Assume $|\alpha_i| \leq 60$ for all i. Since P is a clockwise circular path, $\sum_i \alpha_i = 360$. Thus, there must be a pair (i, j) with $\alpha_i = \alpha_j = 60$ and $\alpha_{i'} = 0$ for $i < i' < j$. This is the case on the path between r's positions in Figures 5a and 5c. One can now show that the tiles at all nodes $v_{i'}$ for $i \leq i' < j$ are moved in direction $d+1$. After the tile at node v_{j-1} is moved, it is not hard to show that lifting the tile at v_j does not disconnect the configuration, see Figure 5c, and that r can return to $B(T \cap \mathcal{T})$ by continuing to apply the LHR on $B_p(T \setminus \mathcal{T})$. ◀

Consequently, moving all m supply tiles to demand nodes takes $\mathcal{O}(mn)$ time steps.

▶ **Theorem 3.3.** *An agent can solve an instance of the Shape Reconfiguration Problem with simply connected target shapes within $\mathcal{O}(mn)$ time steps.*

Existing shape formation algorithms typically have runtimes of $\mathcal{O}(n^2)$ or $\mathcal{O}(nD)$ where D is the diameter of the configuration [9, 10]. Thus, due to the robot's ability to distinguish target from non-target nodes, our algorithm is faster for $m = o(n)$, i.e., whenever the input and target shape largely overlap. In fact, the algorithm is worst-case optimal as some instances cannot be solved in fewer than $\Omega(mn)$ time steps; see, e.g., Figure 7.

▶ **Theorem 3.4.** *Any agent requires $\Omega(mn)$ time steps to solve the Shape Reconfiguration Problem in general.*

4 Conclusion and Outlook

We have shown that a single agent can solve the Shape Reconfiguration Problem for simply connected target shapes in worst-case optimal $\mathcal{O}(mn)$ time steps. In the full version [7], we additionally present an $\mathcal{O}(n^4)$ algorithm for scaled target shapes with holes, which can be adjusted to solve the Shape Reconfiguration Problem for arbitrary target shapes in $\mathcal{O}(mn^3)$ time steps if the agent is equipped with two pebbles.

It remains an open question whether the Shape Reconfiguration Problem can be solved in general without pebbles. We believe that this is not the case, just like exploring a grid maze is impossible for a finite automaton [2, 12]. Another natural follow-up is to examine the Shape Reconfiguration Problem for multiple agents.

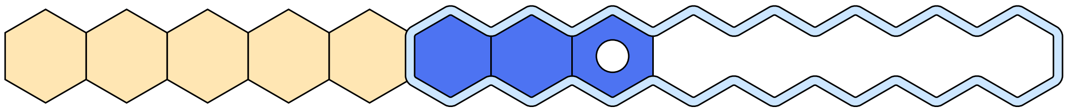


Figure 7 The agent requires $\Omega(mn)$ time steps to move all $m = 5$ supply tiles to demand nodes.

References

1 Aaron T. Becker, Sándor P. Fekete, Li Huang, Phillip Keldenich, Linda Kleist, Dominik Krupke, Christian Rieck, and Arne Schmidt. Targeted drug delivery: Algorithmic methods for collecting a swarm of particles with uniform, external forces. In *International Conference on Robotics and Automation (ICRA)*, pages 2508–2514, 2020. `doi:10.1109/ICRA40945.2020.9196551`.

2 Lothar Budach. Automata and labyrinths. *Mathematische Nachrichten*, 86(1):195–282, 1978. `doi:10.1002/mana.19780860120`.

3 Ho-Lin Chen, David Doty, Dhiraj Holden, Chris Thachuk, Damien Woods, and Chun-Tao Yang. Fast algorithmic self-assembly of simple shapes using random agitation. In *DNA Computing and Molecular Programming (DNA)*, pages 20–36, 2014. `doi:10.1007/978-3-319-11295-4_2`.

4 Gianlorenzo D'Angelo, Mattia D'Emidio, Shantanu Das, Alfredo Navarra, and Giuseppe Prencipe. Asynchronous silent programmable matter achieves leader election and compaction. *IEEE Access*, 8:207619–207634, 2020. `doi:10.1109/ACCESS.2020.3038174`.

5 Joshua J. Daymude, Kristian Hinnenthal, Andréa W. Richa, and Christian Scheideler. Computing by programmable particles. *Distributed Computing by Mobile Entities: Current Research in Moving and Computing*, pages 615–681, 2019. `doi:10.1007/978-3-030-11072-7_22`.

6 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Universal shape formation for programmable matter. In *Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 289–299, 2016. `doi:10.1145/2935764.2935784`.

7 Jonas Friemel, David Liedtke, and Christian Scheffer. Efficient shape reconfiguration by hybrid programmable matter, 2025. `arXiv:2501.08663`.

8 Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Fabian Kuhn, Dorian Rudolph, and Christian Scheideler. Shape recognition by a finite automaton robot. In *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 52:1–52:15, 2018. `doi:10.4230/LIPIcs.MFCS.2018.52`.

9 Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Fabian Kuhn, Dorian Rudolph, Christian Scheideler, and Thim Strothmann. Forming tile shapes with simple robots. *Natural Computing*, 19(2):375–390, 2020. `doi:10.1007/s11047-019-09774-2`.

10 Kristian Hinnenthal, David Liedtke, and Christian Scheideler. Efficient shape formation by 3D hybrid programmable matter: An algorithm for low diameter intermediate structures. In *Symposium on Algorithmic Foundations of Dynamic Networks (SAND)*, pages 15:1–15:20, 2024. `doi:10.4230/LIPICS.SAND.2024.15`.

11 Kristian Hinnenthal, Dorian Rudolph, and Christian Scheideler. Shape formation in a three-dimensional model for hybrid programmable matter. In *European Workshop on Computational Geometry (EuroCG)*, pages 50:1–50:9, 2020. URL: `https://www1.pub.informatik.uni-wuerzburg.de/eurocg2020/data/uploads/papers/eurocg20_paper_50.pdf`.

12 Frank Hoffmann. One pebble does not suffice to search plane labyrinths. In *International Conference on Fundamentals of Computation Theory (FCT)*, pages 433–444, 1981. `doi:10.1007/3-540-10854-8_47`.

13 Benjamin Jenett, Christine Gregg, Daniel Cellucci, and Kenneth Cheung. Design of multifunctional hierarchical space structures. In *Aerospace Conference*, pages 1–10, 2017. doi:10.1109/AERO.2017.7943913.

14 Irina Kostitsyna, David Liedtke, and Christian Scheideler. Universal coating by 3D hybrid programmable matter. In *International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 384–401, 2024. doi:10.1007/978-3-031-60603-8_21.

15 Irina Kostitsyna, Tom Peters, and Bettina Speckmann. Fast reconfiguration for programmable matter. In *International Symposium on Distributed Computing (DISC)*, pages 27:1–27:21, 2023. doi:10.4230/LIPIcs.DISC.2023.27.

16 Irina Kostitsyna, Christian Scheideler, and Daniel Warner. Fault-tolerant shape formation in the amoebot model. In *International Conference on DNA Computing and Molecular Programming (DNA)*, pages 9:1–9:22, 2022. doi:10.4230/LIPICS.DNA.28.9.

17 Alfredo Navarra and Francesco Piselli. Asynchronous silent programmable matter: Line formation. In *Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 598–612, 2023. doi:10.1007/978-3-031-44274-2_44.

18 Nooshin Nokhanji, Paola Flocchini, and Nicola Santoro. Dynamic line maintenance by hybrid programmable matter. *International Journal of Networking and Computing*, 13(1):18–47, 2023. doi:10.15803/ijnc.13.1_18.

19 Matthew J. Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, 13(2):195–224, 2014. doi:10.1007/s11047-013-9379-4.

20 Tommaso Toffoli and Norman Margolus. Programmable matter: Concepts and realization. *Physica D: Nonlinear Phenomena*, 47(1):263–272, 1991. doi:10.1016/0167-2789(91)90296-L.

21 Erik Winfree. *Algorithmic self-assembly of DNA*. PhD thesis, California Institute of Technology, Pasadena, CA, USA, 1998. doi:10.7907/HBBV-PF79.

22 Damien Woods, Ho-Lin Chen, Scott Goodfriend, Nadine Dabby, Erik Winfree, and Peng Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *Innovations in Theoretical Computer Science (ITCS)*, pages 353–354, 2013. doi:10.1145/2422436.2422476.

Computing Optimal Solutions for Chromatic Art Gallery Problems*

Phillip Keldenich[1], Dominik Krupke[1], and Jan Siemsen[1]

1 **Algorithms Division, TU Braunschweig**
`{keldenich,krupke}@ibr.cs.tu-bs.de`, `j.siemsen@tu-bs.de`

Abstract

We study exact algorithms for two chromatic variants of the Art Gallery Problem, motivated by radio frequency signal interference. For the Chromatic AGP with vertex guards, we demonstrate that a SAT-based algorithm can solve instances with up to 40,000 vertices, vastly outperforming a previous MIP-based approach. We also present SAT- and MIP-based algorithms for the Conflict-Free Chromatic AGP with vertex guards, solving instances with up to 2500 vertices to optimality. Our formulation works unchanged for instances with holes. However, our empirical evaluation indicates that introducing holes renders instances with as few as 300 vertices and 30 holes challenging for the Chromatic AGP.

1 Introduction

Achieving comprehensive indoor signal coverage is vital, yet overlapping signals can cause interference. To avoid interference, one may assign distinct frequencies in overlapping areas (*chromatic* assignment) or ensure at least one unique frequency at each point (*conflict-free* assignment); these problems are usually modeled as graph or hypergraph coloring problems. Covering a polygon using the minimum number of guards gives rise to the Art Gallery Problem (AGP). Combining these problems by modeling the range of a base station g as straight-line visibility polygon $\mathcal{V}(g)$ in a given (possibly non-simple) polygon P leads to the Chromatic Art Gallery Problem (CAGP) and the Conflict-Free Chromatic AGP (CFCAGP), see Fig. 1 for examples. Given a polygon P with vertex set V, we seek a set of (vertex) *guards* $S \subseteq V$ and a coloring $c : S \rightarrow \mathbb{N}$ with a minimum number K of colors such that $\bigcup_{g\in S} \mathcal{V}(g) = P$ and:

CAGP if $p \in P$ sees guards $g_1 \neq g_2$, then $c(g_1) \neq c(g_2)$,

CFCAGP for each $p \in P$, some guard g for p has a unique color among all guards seeing p.

We study exact algorithms for these two problems using geometric insights to discretize the problems, and then employ SAT and MIP solvers to find optimal solutions. Our empirical study shows that the SAT-based approach for the CAGP significantly outperforms the MIP-based method of Zambon et al. [13], enabling us to solve instances with up to 40,000 vertices (see Fig. 2), compared to the previous limit of 2500 vertices.

We also introduce new formulations for the CFCAGP, which has not been previously explored from a practical standpoint and appears to be considerably more challenging. Despite the increased complexity, all tested random simple polygons with up to 2500 vertices (see Fig. 3) were solved within 600 s on commodity hardware using SAT solvers. Introducing holes raises the difficulty, leaving some instances with 300 vertices and 30 holes unsolved.

* This work was supported by DFG project "Computational Geometry: Solving Hard Optimization Problems" (CG:SHOP), FE407/21-1.

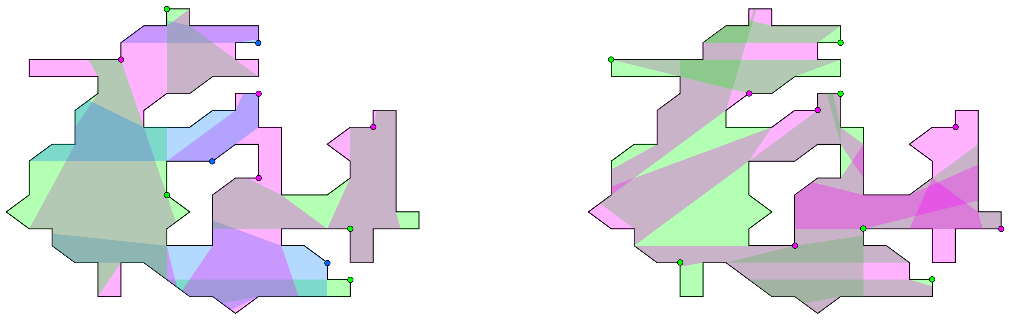


Figure 1 Example for the CAGP with 3 colors (left), and for the CFCAGP with 2 colors (right).

Related Work In [7, 8], it was shown that the CAGP is NP-hard for $K \geq 2$ colors in non-simple polygons and can require up to $\Theta(n)$ colors in simple polygons on n vertices; this holds even if one is not restricted to vertex guards. In [9], NP-hardness is also established for $K \geq 2$ in orthogonal polygons. On the positive side, [5] proves that, given a polygon P and a guard set S, an optimum coloring of S that assigns distinct colors to every pair of guards with intersecting visibility regions can be computed in polynomial time. Furthermore, [8] presents a polynomial-time algorithm for finding a 2-colorable guard set in a simple polygon if one exists, given a discrete set of candidate guard locations; the authors also present a $\mathcal{O}(\log(\chi_G(P)))$-approximation algorithm. A linear-time 6-approximation for simple orthogonal polygons and an exact algorithm for histogram polygons appear in [9].

Lower and upper bounds on the required number of colors have been extensively studied. In [6], it was shown that for $K \geq 3$, there is a polygon with $4K$ vertices requiring at least K colors, and a strictly monotone polygon with $3K^2$ vertices also requiring K colors. For odd $K \geq 3$, the same work constructs a monotone orthogonal polygon R_K with $4K^2 + 10K + 10$ vertices and $\chi_G(R_K) \geq K$. Meanwhile, in [1], it was shown that for $K \geq 1$, there is a polygon with $3K$ vertices needing at least $K/2$ colors, and a monotone orthogonal polygon with $4K^2$ vertices needing at least $K/4$ colors. Regarding upper bounds, in [6] it was shown any spiral polygon can be colored with at most 2 colors, and any staircase polygon with at most 3.

The only practical work on the CAGP so far is [13], which proposes a MIP formulation which is the foundation for our work.

For the CFCAGP, [10] proves NP-hardness for two colors in non-simple polygons, while [1] shows that $\mathcal{O}(\log n)$ colors suffice for monotone polygons and $\mathcal{O}(\log^2 n)$ for general polygons. To our knowledge, no prior work addresses practical solutions to the CFCAGP.

2 Preliminaries

Consider a (possibly non-simple) polygon P with vertex set V. We seek to cover P using guards placed at a subset of vertices $S \subseteq V$, where each guard $g \in S$ covers its visibility polygon $\mathcal{V}(g)$. We say that a set S with $\bigcup_{g \in S} \mathcal{V}(g) = P$ is a *guard set*. Conversely, a set of points W is called a *witness set* if covering W with vertex guards guarantees that all of P is covered. In other words, W is a witness set if, for any $S' \subseteq V$, $W \subseteq \bigcup_{g \in S'} \mathcal{V}(g)$ implies $\bigcup_{g \in S'} \mathcal{V}(g) = P$. Points in a witness set are *witnesses*. We derive finite witness sets by partitioning P into *atomic visibility polygons* (AVPs).

▶ **Definition 2.1.** Overlaying the visibility polygons $\mathcal{V}(g)$ for all $g \in V$ produces a planar arrangement. Each face of this arrangement is an *atomic visibility polygon (AVP)*.

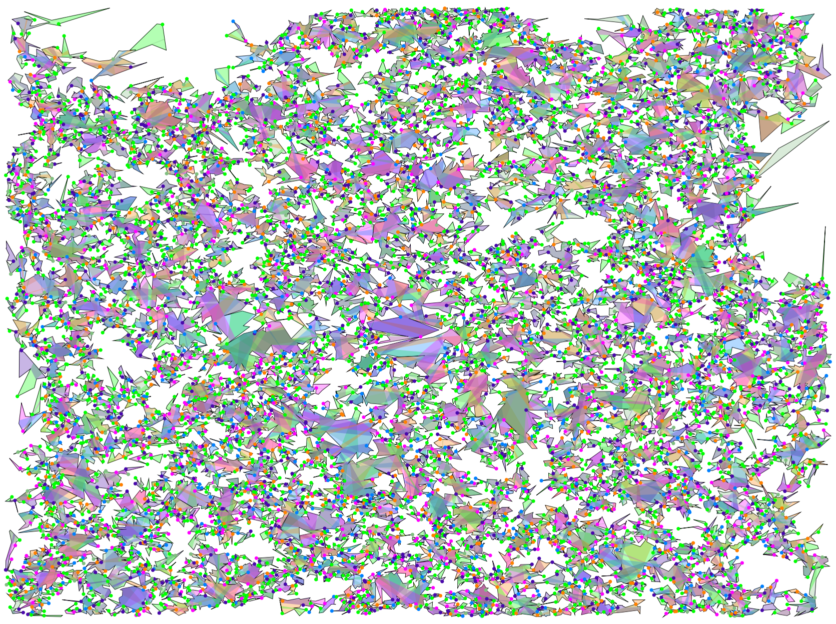


Figure 2 Optimal solution for an instance with 40,000 vertices of the CAGP.

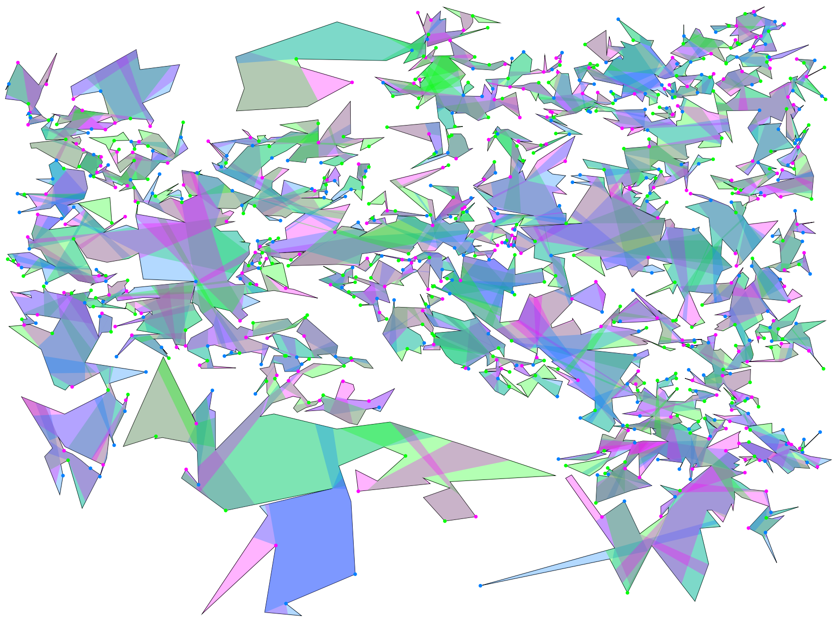


Figure 3 Optimal solution for an instance with 2500 vertices of the CFCAGP.

Let $\mathcal{F}$ be the set of all AVPs. Because visibility remains the same within each face, if a guard g covers any point in the interior of an AVP $f \in \mathcal{F}$, then g covers every point of f. For each $f \in \mathcal{F}$, define $V_f = \{g \in V \mid \mathcal{V}(g) \supseteq f\}$. We introduce a partial order $\prec$ on $\mathcal{F}$ where $f \prec f'$ if and only if $V_f \subset V_{f'}$. An AVP f is called a *shadow* AVP if it is minimal in $(\mathcal{F}, \prec)$, and a *light* AVP if it is maximal; see Fig. 4.

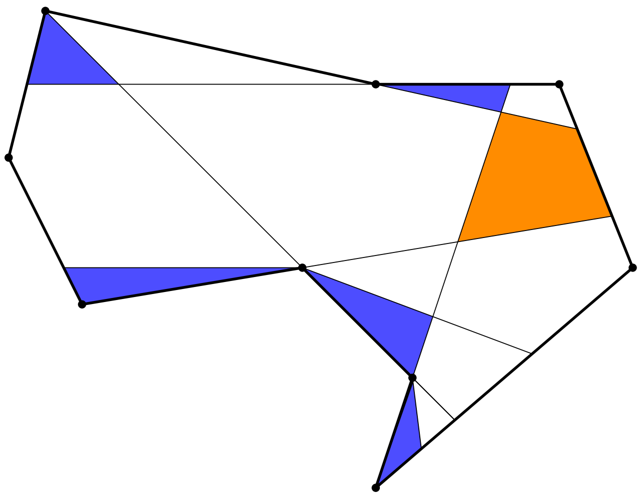


Figure 4 AVP arrangement with shadow AVPs (in blue) and light AVPs (in orange).

By placing one witness in the interior of each shadow AVP, we obtain a *shadow witness set* W. Let V_w equal V_f for shadow witness $w \in W$ in shadow AVP $f \in \mathcal{F}$. It was shown in [3] that $S' \subseteq V$ is a guard set if and only if S' covers all witnesses in any shadow witness set W, i.e., $\forall w \in W : V_w \cap S' \neq \emptyset$. We compute the AVPs by first calculating the visibility polygon for each vertex, then recursively dividing the list of arrangements in halves and overlaying them using CGAL [11, 12].

3 Formulations

Here, we describe the MIP and SAT formulations for both the CAGP and CFCAGP.

3.1 Chromatic AGP

Our CAGP formulations use the *2-link visibility graph* $G_{\text{vis}} = (V, E_{\text{vis}})$ that has an edge $\{g, h\} \in E_{\text{vis}}$ exactly when $\mathcal{V}(g) \cap \mathcal{V}(h) \neq \emptyset$. The graph G_{vis} is sufficient to model the coloring constraints and can be obtained efficiently from the AVPs because the V_f of the light AVPs $f \in \mathcal{F}$ yield all intersections.

MIP formulation As a MIP formulation, we use the approach by Zambon et al. [13].

SAT formulation We also use the following SAT formulation to model whether K colors suffice; this formulation is then used in a binary search to identify the minimum number of colors. For each color $k \in \{1, \dots, K\}$ and each guard $g \in V$, we have a variable x_{gk} indicating whether g is used with color k. For each edge gh in G_{vis} and each color k, the clause $\neg x_{gk} \vee \neg x_{hk}$ ensures guard colors are distinct. For each shadow witness w in an arbitrary shadow witness set W, we include the clause

$$\bigvee_{g\in V_w, k\in\{1,\dots,K\}} x_{gk} \tag{1}$$

to ensure coverage. Finally, for each pair $i \neq j$ of colors and each guard g, the clauses $\neg x_{gi} \vee \neg x_{gj}$ ensure that each guard has at most one color.

Lazy Constraints While we can restrict ourselves to shadow witnesses for the CAGP, the number of shadow witnesses can still be quite large. We therefore initially limit ourselves to the $|V|$ shadow witnesses w with smallest V_w. In the MIP approach, we detect and add possible missing witnesses using Gurobi's callback interface when we encounter an integral feasible solution. In the SAT approach, we detect and add missing witnesses whenever we find an optimal solution for the current witness set. The incrementality of the SAT solvers supported by PySAT means that the ensuing solve is usually much cheaper than a fresh start.

3.2 Conflict-Free Chromatic AGP

For modeling conflict-free coloring, we unfortunately cannot just use G_{vis}, but instead must work with all AVPs in $\mathcal{F}$. By enforcing that for every AVP $f \in \mathcal{F}$ there is a guard with a unique color in V_f, we implicitly ensure coverage.

MIP formulation Our MIP formulation of the CFCAGP introduces a binary variable $c_k \in \{0,1\}$ for each color $k \in \{1,\dots,K\}$ to indicate whether color k is used, and a binary variable $x_{gk} \in \{0,1\}$ for each guard $g \in V$ to indicate whether g is assigned color k. Additionally, we introduce a binary variable $u_{fk} \in \{0,1\}$ for each AVP $f \in \mathcal{F}$ and color k, indicating whether there is exactly one guard $g \in V_f$ guarding f with color k. The constraints

$$\sum_{g\in V} x_{gk} \geq c_k, \quad \sum_{g\in V} x_{gk} \leq |V|\cdot c_k \quad \forall k \in \{1,\dots,K\} \tag{2}$$

ensure that $c_k = 1$ if and only if color k is used. The constraints

$$\sum_{g\in V_f} x_{gk} \geq u_{fk}, \quad \sum_{g\in V_f} x_{gk} \leq 1 + |V_f|\cdot(1-u_{fk}) \quad \forall f \in \mathcal{F}, k \in \{1,\dots,K\} \tag{3}$$

enforce that $u_{fk} = 1$ precisely if color k occurs exactly once in V_f. In addition,

$$\sum_{k=1}^{K} u_{fk} \geq 1, \quad \sum_{k=1}^{K} x_{gk} \leq 1 \quad \forall f \in \mathcal{F} \tag{4}$$

ensures each AVP $f \in \mathcal{F}$ is guarded by at least one uniquely colored guard and that each guard is assigned at most one color. Finally, for each $k < K$, we optionally add the following symmetry-breaking constraints:

$$c_k \geq c_{k+1}, \quad \sum_{g\in V} x_{gk} \geq \sum_{g\in V} x_{g(k+1)}. \tag{5}$$

SAT formulation We also present a SAT formulation to test whether K colors suffice for the CFCAGP. This formulation uses binary variables x_{gk} and u_{fk}, defined analogously to the MIP, where $g \in V$, $k \in \{1, \dots, K\}$, and $f \in \mathcal{F}$. For each AVP f and color k, the clauses

$$\neg u_{fk} \vee \bigvee_{g \in V_f} x_{gk}, \qquad \bigwedge_{(i,j) \in \binom{V_f}{2}} \left(\neg u_{fk} \vee \neg x_{ik} \vee \neg x_{jk}\right) \quad \forall f \in \mathcal{F}, k \in \{1, \dots, K\} \tag{6}$$

ensure u_{fk} can only be true if exactly one guard in V_f is colored k. Note that, compared to the MIP formulation, where a single constraint with $O(|V_f|)$ non-zeros suffices to enforce that at most one guard in V_f can have color k if u_{fk} is true, we use $O(|V_f|^2)$ three-element clauses to enforce this, making the formulation slightly less succinct. To guarantee conflict-free coverage for each AVP, the clause $\bigvee_{k \in \{1,\dots,K\}} u_{fk}$ is added for every $f \in \mathcal{F}$. Finally, to enforce that each guard can be assigned at most one color, the clause $\neg x_{gi} \vee \neg x_{gj}$ is included for all guards $g \in V$ and every pair of distinct colors $i \neq j$.

4 Experimental evaluation

In the following, we evaluate how the approaches perform on benchmark instances. All experiments are performed on WSL2 (Windows 11, version 22H2) using an AMD Ryzen 7 7800X3D with 28 GB of RAM. Geometric operations are implemented using CGAL [11]; all solvers are implemented in Python 3.10.14, using Gurobi version 11.0.1 as MIP solver and PySAT version 0.1.8.dev9 for SAT solvers. All solvers run with a time limit of 600 s, excluding the time to geometrically process the instances.

Deciding for a SAT Solver Unlike the MIP field, dominated by a few commercial solvers, the SAT solver landscape remains diverse, with heterogeneous performance across instances and no clear leader. We begin by comparing different SAT solvers on randomly generated simple polygons [2] with 100–2500 vertices for CAGP and polygons with holes with 100–300 vertices for CFCAGP to select the most suitable solvers for subsequent experiments. All evaluated SAT solvers are provided by PySAT with a uniform interface.

For the CAGP, we consider two SAT formulation variants: *version 1*, which allows guards to have redundant additional colors, and *version 2*, which enforces exactly one color. As shown in Fig. 5, CaDiCaL103 performs best on both versions. In contrast, for the CFCAGP, Fig. 6 reveals that Minisat22 and Minicard excel. Perhaps surprisingly, CaDiCaL103 is weak for CFCAGP, while Minisat22 and Minicard underperform for CAGP.

Because all solvers are single-threaded, we build a portfolio to run them in parallel, stopping as soon as one solver finishes. For the CAGP, we use CaDiCaL103 and Glucose42 with both versions, and Glucose4 with version 2. For the CFCAGP, our portfolio includes Minisat22, Minicard, Gluecard 3, Gluecard 4, and Glucose 3.

CAGP We now compare our SAT-based approach with the MIP-based method of Zambon et al. [13] on polygons with and without holes. As shown in Fig. 7, our SAT-based method nearly instantly solves the benchmark instances without holes from [2, 13]. Although the MIP approach benefits from modern hardware and a recent Gurobi version, it still takes considerably longer than our SAT-based method to solve the same instances. For polygons with holes, which we also take from [2], neither approach solves all instances (smallest unsolved instance having 300 vertices), but the SAT method visibly solves more within the time limit, as seen in Fig. 8. A more thorough investigation of this observed increase in practical difficulty is a possible direction for future research. To evaluate scalability, we used

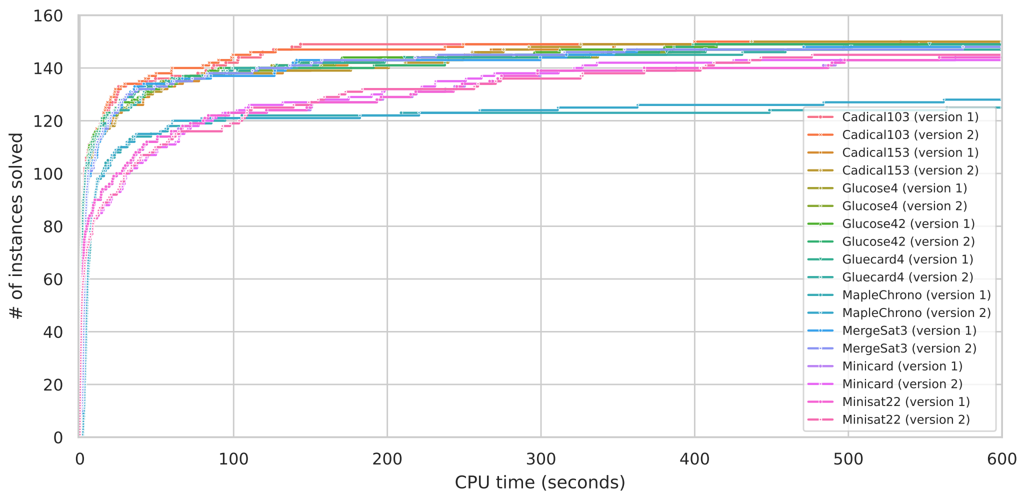


Figure 5 CAGP instances solved over time for different PySAT backends (higher is better).

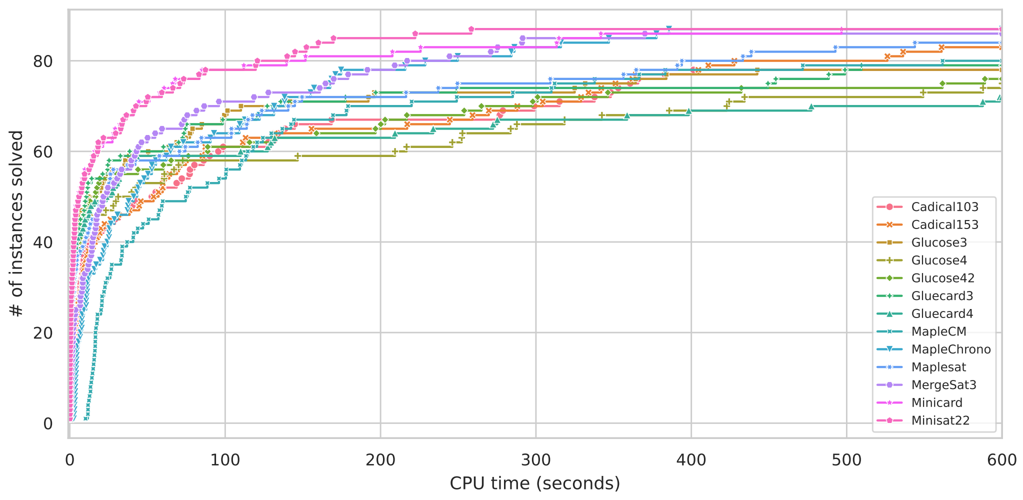


Figure 6 CFCAGP instances solved over time for different PySAT backends (higher is better).

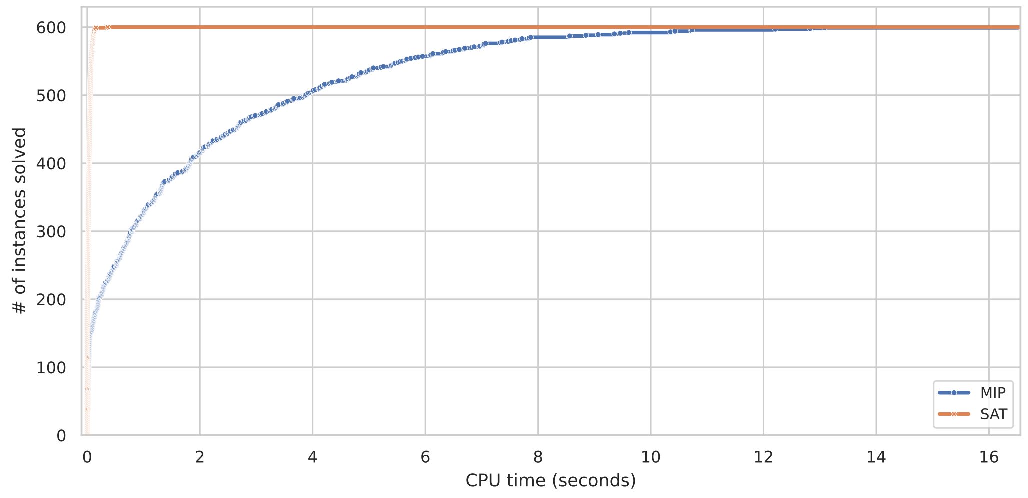


Figure 7 CAGP instances solved over time (higher is better); the SAT-based approach vastly outperforms the MIP. Both approaches could solve all instances within the time limit.

larger instances generated by the *fast polygon generator* (fpg) from the Salzburg Database [4]. The largest instance the MIP approach solved in time had 9000 vertices, taking 181.19 s. Our SAT method solved the same instance in 1.7 s and handled instances up to 40,000 vertices, solving one in 156.29 s. Beyond this size, memory consumption becomes a limiting factor.

CFCAGP Finally, let us compare the performance of the SAT and MIP approaches for the CFCAGP. On the simple polygons from [2], the SAT approach again outperforms the MIP approach (see Fig. 9), which times out on instances with 60 vertices that took the SAT approach less than 1 s. This was also true on the *fpg* instances; here, the SAT solver solved instances with up to 2500 vertices. For larger instances, we run out of memory during geometric preprocessing and the construction of the SAT model.

5 Conclusion

We empirically evaluated exact algorithms for the Chromatic Art Gallery Problem (CAGP) and the Conflict-Free Chromatic Art Gallery Problem (CFCAGP) using both MIP and SAT solvers. Our SAT-based approach solved CAGP instances with up to 40,000 vertices, surpassing the 2500 vertex limit achieved by Zambon et al. [13] in 2014. For the CFCAGP, we successfully solved instances with up to 2500 vertices. Despite the popularity and sophistication of modern MIP solvers, our findings show that simpler SAT solvers perform significantly better on these problems, highlighting the importance of selecting the right technology in optimization.

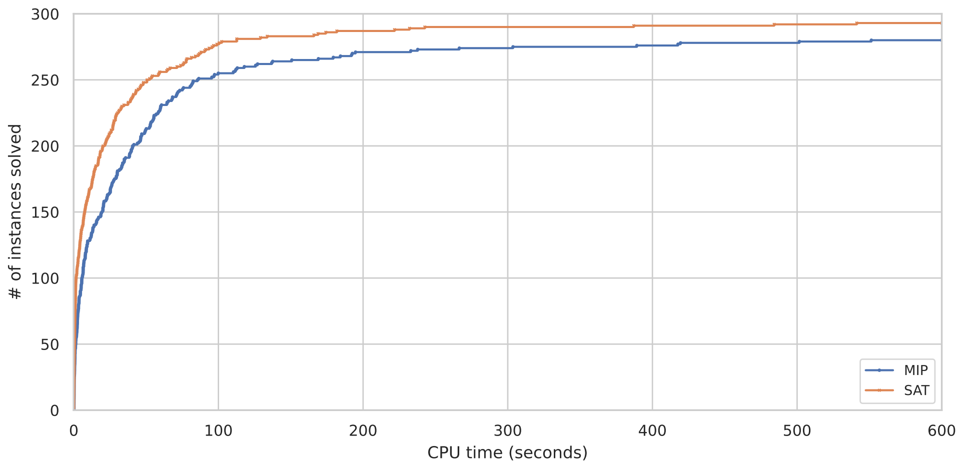


Figure 8 Number of CAGP instances with holes solved over time (higher is better).

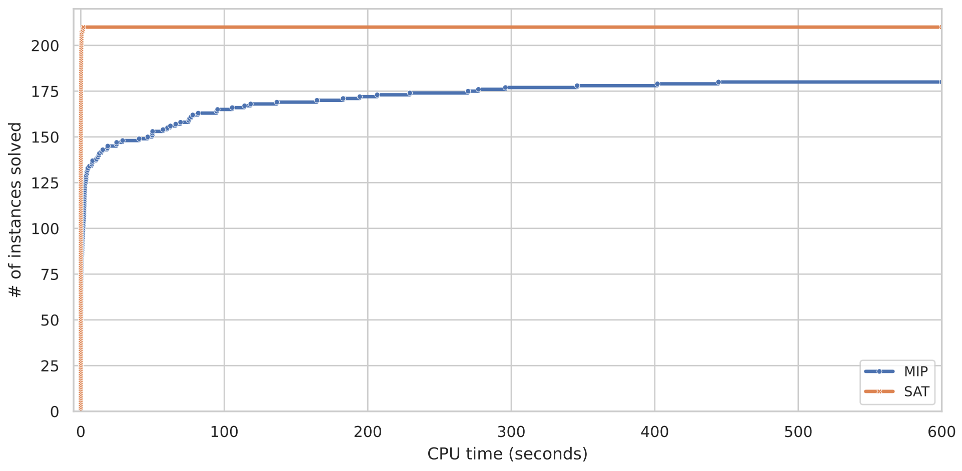


Figure 9 Number of CFCAGP instances solved over time; instances limited to 300 vertices due to poor MIP performance. SAT solves all instances from the full benchmark set in less than 6 s.

References

1 Andreas Bärtschi. *Coloring variations of the art gallery problem.* PhD thesis, Master's thesis, Department of Mathematics, ETH Zürich, 2011.

2 Marcelo C. Couto, Pedro J. de Rezende, and Cid C. de Souza. Instances for the Art Gallery Problem, 2009. www.ic.unicamp.br/∼cid/Problem-instances/Art-Gallery.

3 Marcelo C. Couto, Pedro J. de Rezende, and Cid C. de Souza. An exact algorithm for minimizing vertex guards on art galleries. *International Transactions in Operational Research*, 18(4):425–448, 2011.

4 Günther Eder, Martin Held, Steinþór Jasonarson, Philipp Mayer, and Peter Palfrader. Salzburg database of polygonal data: Polygons and their generators. *Data in Brief*, 31:105984, 2020.

5 Lawrence H. Erickson and Steven M. LaValle. How many landmark colors are needed to avoid confusion in a polygon? In *2011 IEEE International Conference on Robotics and Automation*, pages 2302–2307. IEEE, 2011.

6 Lawrence H. Erickson and Steven M. LaValle. An art gallery approach to ensuring that landmarks are distinguishable. In *Robotics: science and systems*, volume 7, pages 81–88, 2012.

7 Sándor P. Fekete, Stephan Friedrichs, and Michael Hemmer. Complexity of the general chromatic art gallery problem. *arXiv preprint arXiv:1403.2972*, 2014.

8 Sándor P. Fekete, Stephan Friedrichs, Michael Hemmer, Joseph BM Mitchell, and Christiane Schmidt. On the chromatic art gallery problem. In *CCCG*, 2014.

9 Hamid Hoorfar and Alireza Bagheri. NP-completeness of chromatic orthogonal art gallery problem. *The Journal of Supercomputing*, 77(3):3077–3109, 2021.

10 Chuzo Iwamoto and Tatsuaki Ibusuki. Vertex-to-point conflict-free chromatic guarding is NP-hard. In *International Conference and Workshops on Algorithms and Computation*, pages 111–122. Springer, 2022.

11 The CGAL Project. *CGAL User and Reference Manual.* CGAL Editorial Board, 6.0.1 edition, 2024. URL: `https://doc.cgal.org/6.0.1/Manual/packages.html`.

12 Ron Wein, Eric Berberich, Efi Fogel, Dan Halperin, Michael Hemmer, Oren Salzman, and Baruch Zukerman. 2D arrangements. In *CGAL User and Reference Manual.* CGAL Editorial Board, 6.0.1 edition, 2024. URL: `https://doc.cgal.org/6.0.1/Manual/packages.html#PkgArrangementOnSurface2`.

13 Maurício J. O. Zambon, Pedro J. de Rezende, and Cid C. de Souza. An exact algorithm for the discrete chromatic art gallery problem. In *International Symposium on Experimental Algorithms*, pages 59–73. Springer, 2014.

Incremental Planar Nearest Neighbor Queries with Optimal Query Time

John Iacono[1] **and Yakov Nekrich**[2]

1 **Université libre de Bruxelles, Belgium**
2 **Michigan Technological University, USA**

Abstract

In this paper we show that two-dimensional nearest neighbor queries can be answered in optimal $O(\log n)$ time while supporting insertions in $O(\log^{1+\varepsilon} n)$ time. No previous data structure was known that supports $O(\log n)$-time queries and polylog-time insertions. In order to achieve logarithmic queries our data structure uses a new technique related to fractional cascading that leverages the inherent geometry of this problem. Our method can be also used in other semi-dynamic scenarios.

1 Introduction

In the nearest neighbor problem a set of points S is stored in a data structure so that for a query point q the point $p \in S$ that is closest to q can be found efficiently. The nearest neighbor problem and its variants are among the most fundamental and extensively studied problems in computational geometry; we refer to e.g. [17] for a survey. In this paper we study dynamic data structures for the Euclidean nearest neighbor problem in two dimensions. We show that the optimal $O(\log n)$ query time for this problem can be achieved while allowing insertions in time $O(\log^{1+\epsilon} n)$.

Previous Work. See Table 1. In the static scenario the planar nearest neighbor problem can be solved in $O(\log n)$ time by point location in Voronoi diagrams. However the dynamic variant of this problem is significantly harder because Voronoi diagrams cannot be dynamized efficiently: it was shown by Allen et al. [2] that a sequence of insertions can lead to $\Omega(\sqrt{n})$ amortized combinatorial changes per insertion in the Voronoi diagram. A static nearest-neighbor data structure can be easily transformed into an insertion-only data structure using the logarithmic method of Bentley and Saxe [3] at the cost of increasing the query time to $O(\log^2 n)$. Several researchers [9, 11, 18] studied the dynamic nearest neighbor problem in the situation when the sequence of updates is random in some sense (e.g. the deletion of any element in the data structure is equally likely). However their results cannot be extended to the case when the complexity of a specific sequence of updates must be analyzed.

Using a lifting transformation [10], 2-d nearest neighbor queries can be reduced to extreme point queries on a 3-d convex hulls. Hence data structures for the dynamic convex hull in 3-d can be used to answer 2-d nearest neighbor queries. The first such data structure (without assumptions about the update sequence) was presented by Agarwal and Matoušek [1]. Their data structure supports queries in $O(\log n)$ time and updates in $O(n^{\varepsilon})$ time; another variant of their data structure supports queries in $O(n^{\varepsilon})$ time and updates in $O(\log n)$ time. A major improvement was achieved in a seminal paper by Chan [4]. The data structure in [4] supports queries in $O(\log^2 n)$ time, insertions in $O(\log^3 n)$ expected time and deletions in $O(\log^6 n)$ expected time. The update procedure can be made deterministic using the result of Chan and Tsakalidis [6]. The deletion time was further reduced to $O(\log^5 n)$ [15] and to $O(\log^4 n)$ [5]. This sequence of papers makes use of shallow cuttings, a general powerful technique, but, alas, all uses of it for the point location problem in 2-d have resulted in $O(\log^2 n)$ query times.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

	Query	**Insert**	**Delete**
Bentley and Saxe 1980 [3]	$O(\log^2 n)$	$O(\log^2 n)$	Not supported
Agarwal and Matoušek 1995 [1]	$O(\log n)$	$O(n^\varepsilon)$	$O(n^\varepsilon)$
''	$O(n^\varepsilon)$	$O(\log n)$	$O(\log n)$
Chan 2010 [4]	$O(\log^2 n)$	$O(\log^3 n)$ †	$O(\log^6 n)$ †
Chan and Tsakalidis 2016 [6]	$O(\log^2 n)$	$O(\log^3 n)$	$O(\log^6 n)$
Kaplan et al. 2020 [15]	$O(\log^2 n)$	$O(\log^3 n)$	$O(\log^5 n)$
Chan 2020 [5]	$O(\log^2 n)$	$O(\log^3 n)$	$O(\log^4 n)$
Here	$O(\log n)$	$O(\log^{1+\varepsilon} n)$	Not supported

Table 1 Known results. Insertion and deletion times are amortized, † denotes in expectation.

Even in the insertion-only scenario, the direct application of the 45-year-old classic technique of Bentley and Saxe [3] remains the best insertion-only method with polylogarithmic update before this work; no data structure with $O(\log^{2-\epsilon} n)$ query time and polylogarithmic update time was described previously for any $\epsilon > 0$.

Our Results. We demonstrate that optimal $O(\log n)$ query time and poly-logarithmic update time can be achieved in some dynamic settings:

1. We describe a *semi-dynamic insertion-only* data structure that uses $O(n)$ space, supports insertions in $O(\log^{1+\varepsilon} n)$ amortized time and answers queries in $O(\log n)$ time.
2. In the *semi-online* scenario, introduced by Dobkin and Suri [12], we know the deletion time of a point p when a point p is inserted, i.e., we know how long a point will remain in a data structure at its insertion time. We describe a semi-online fully-dynamic data structure that answers queries in $O(\log n)$ time and supports updates in $O(\log^{1+\varepsilon} n)$ amortized time. The same result is also valid in the *offline* scenario when the entire sequence of updates is known in advance.
3. In the *offline partially persistent scenario*, the sequence of updates is known and every update creates a new version of the data structure. Queries can be asked to any version of the data structure. We describe an offline partially persistent data structure that uses $O(n \log^{1+\varepsilon} n)$ space and construction time and answers queries in $O(\log n)$ time.

All three problems considered in this paper can be reduced to answering point location queries in (static) Voronoi diagrams of $O(\log n)$ different point sets. For example, we can obtain an insertion-only data structure by using the logarithmic method of Bentley and Saxe [3], which we now briefly describe. The input set S is partitioned into a logarithmic number of subsets $S_1, \ldots, S_f$ of exponentially increasing sizes. In order to find the nearest neighbor of some query point q we locate q in the Voronoi diagram of each set S_i and report the point closest to q among these nearest neighbors. Since each point location query takes $O(\log n)$ time, answering a logarithmic number of queries takes $O(\log^2 n)$ time.

The fractional cascading technique [7] applied to this problem in one dimension decreases the query cost to logarithmic by sampling elements of each S_i and storing copies of the sampled elements in other sets S_j, $j < i$. Unfortunately, it was shown by Chazelle and Liu [8] that fractional cascading does not work well for two-dimensional non-orthogonal problems, such as point location: in order to answer $O(\log n)$ point location queries in $O(\log n)$ time, we would need $\tilde{\Omega}(n^2)$ space, even in the static scenario.

To summarize, the two obvious approaches to the insertion-only problem are to maintain a single search structure and update it with each insertion, the second is to maintain a collection of static Voronoi diagrams of exponentially-increasing size and to execute nearest neighbor queries by finding the closest point in all structures, perhaps aided by some kind of fractional cascading. The first approach cannot obtain polylogarithmic insertion time due to the lower bound on the complexity change in Voronoi diagrams caused by insertions [2], and the second approach cannot obtain $O(\log n)$ search time due to Chazelle and Liu's lower bound [8]. Our main intellectual contribution is showing that the lower bound of Chazelle and Liu [8] can be circumvented for the case of point location in Voronoi diagrams. Specifically, a strict fractional cascading approach requires finding the closest point to a query point in each of the subsets S_i; we loosen this requirement: in each S_i, we either find the closest point or provide a certificate that the closest point in S_i is not the closest point in S. This new, powerful and more flexible form of fractional cascading is done by using a number of novel observations about the geometry of the problem. We imagine our general technique may be applicable to speeding up search in other dynamic search problems.

2 Overview of method

Here we provide a high-level overview of our method, with details deferred to the full version. We let S denote the set of points currently stored in the structure, and use n to denote $|S|$. Let $\mathcal{S} = \{S_1, S_2, \ldots S_f\}$ denote a partition of S into sets of exponentially-increasing size where $f := |\mathcal{S}| = \Theta(\log n)$ and $|S_i| = \Theta(2^i)$. Let $NN(P,q)$ be the nearest neighbor of q in a point set P. Given a point q, the computation of $NN(S,q)$ is the query that our data structure will support.

We now define a sequence of point sets $T_1, \ldots T_f$. The intuition is that, as in classical fractional cascading [7], the set T_i contains all elements of S_i and a sample of elements from the sets T_j where $j > i$; this implies the last sets are equal: $T_f = S_f$. This sampling will be provided by the function $Sample_j(k)$ which returns a subset of T_j of size $O(|T_j|/2^{2k})$; while it will have other important properties, for now only the size matters.

We now can formally define T_i: $T_i := S_i \cup \bigcup_{j=i+1}^{f} Sample_j(j-i)$. From this definition we have several observations: (i) $T_f = S_f$, (ii) T_i is a function of the S_j, for $j \geq i$, (iii) $S = \cup_{i=1}^{f} T_i$, (iv) $NN(S,q) \in \bigcup_{i=1}^{f}\{NN(T_i,q)\}$, (v) $|T_i| = \Theta(2^i)$, and (vi) For any i $\sum_{j=i+1}^{f} |Sample_j(j-i)| = \Theta(|T_i|)$.

Voronoi and Delaunay. Let $Vor(P)$ be the Voronoi diagram of point set P, let $Cell(P,p)$ be the cell of a point p in $Vor(P)$, that is the locus of points in the plane whose closest element in P is p. Thus $q \in Cell(P,p)$ is equivalent to $NN(P,q) = p$. Let $|Cell(P,p)|$ be the complexity of the cell, that is, the number of edges on its boundary. Let $G(P)$ refer to the Delaunay graph of P, the dual graph of the Voronoi diagram of P; the degree of p in $G(P)$ is thus $|Cell(P,p)|$ and each point in P corresponds to a unique vertex in $G(P)$. We will find it useful to have a compact notation for expressing the union of Voronoi cells; thus for a set of points $P' \subseteq P$, let $Cells(P,P')$ denote $\bigcup_{p\in P'} Cell(P,p)$.

Pieces and Fringes. Given a graph, $G = (V,E)$, and a set of vertices $V' \subseteq V$, the *fringe* of V' (with respect to G) is the subset of V' incident to edges whose other endpoint is in $V \setminus V'$. Let $G = (V,E)$ be a planar graph. For any r, Frederickson [13] showed the vertices of G can be decomposed[1] into $\Theta(|V|/r)$ *pieces*, so that: (i) Each vertex is in at

[1] We use the word *decomposed* to mean a division of a set into into a collection sets, the *decomposition*, whose union is the original set, but, unlike with a partition, elements may belong to multiple sets.

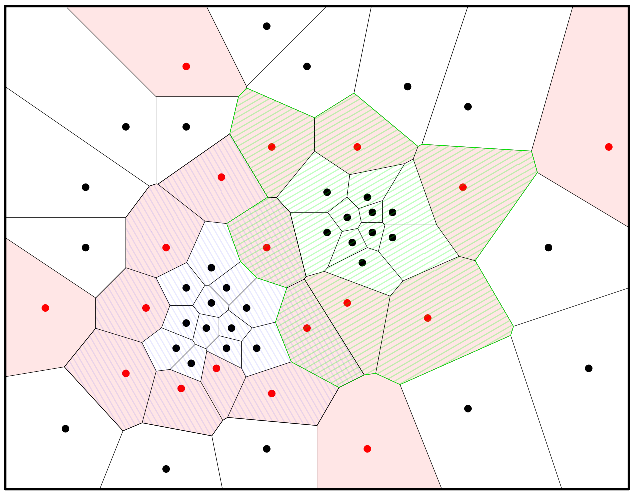


Figure 1 Part of a Voronoi diagram for a point set T_j. Two elements of $Pieces_j(k)$ have been highlighted, one in striped blue, call it $Piece_j^1(k)$, and one in striped green, call it $Piece_j^2(k)$. For each piece, the cells of fringe vertices are shaded red. Thus, the set $Sample_j(k)$ are the red verticies, and the region $Cells(T_j, Sample_j(k))$ is shaded red. The green-and-red shaded region is $Cells(T_j, Sep_j^2(k))$ and the green-but-not-red shaded region is $Cells(T_j, \overline{Sep}_j^2(k))$

.

least one piece. (ii) Each piece has at most r vertices in total and only $O(\sqrt{r})$ vertices on its fringe. (iii) If a vertex is a non-fringe vertex of a piece (with respect to G), then it is not in any other pieces. (iv) The total size of all pieces is in $\Theta(|V|)$. Intuitively, the pieces are almost a partition of V where those vertices on the fringe of each piece may appear in multiple pieces. Such a decomposition of G can be computed in time $O(|V|)$ [14, 16]. We will apply this decomposition to T_i, which is both a point set and the vertex set of $G(T_i)$, for exponentially increasing sizes of r.

Given integers $1 \leq k < j < f$, let $Pieces_j(k) := \{Piece_j^1(k), \dots Piece_j^{|Pieces_j(k)|}(k)\}$ be a decomposition of T_j into $r = \Theta(|T_j|/2^{4k})$ subsets such that each subset $Piece_j^l(k)$ has size $O(2^{4k})$ and a fringe of size $O(2^{2k})$ with respect to $G(T_i)$. We let $Seps_j(k) := \{Sep_j^1(k), \dots Sep_j^{|Pieces_j(k)|}(k)\}$ be defined so that $Sep_j^\ell(k)$ denotes the fringe of $Piece_j^\ell(k)$, and let $\overline{Sep}_j^\ell(k)$ be $Piece_j^\ell(k) \setminus Sep_j^\ell(k)$. Thus each $Piece_j^\ell(k)$ is partitioned into its fringe vertices, $Sep_j^\ell(k)$, and its interior non-fringe vertices $\overline{Sep}_j^\ell(k)$; note that $\overline{Sep}_j^\ell(k)$ may be empty if all elements of $Piece_j^\ell(k)$ are on the fringe. Finally, we define $Sample_j(k)$ to be the union of all the fringe vertices: $Sample_j(k) := \bigcup_{Sep \in Seps_j(k)} Sep$. Thus, $Seps_j(k)$ is a partition decomposition of $Sample_j(k)$.

For any $k \in [1..j-1]$, the decomposition of T_j into $Pieces_j(k)$, the partition of each $Piece_j^\ell(k)$ into $Sep_j^\ell(k)$ and $\overline{Sep}_j^\ell(k)$, and the set $Seps_j(k)$ can all be computed in time $O(|T_j|)$ using [16] if the Delaunay triangulation is available; if not it can be computed in time $O(|T_j| \log |T_j|)$. Thus computing these for all valid i takes time and space $O(|T_j| \log^2 |T_j|)$ as $k < j = O(\log |T_j|)$.

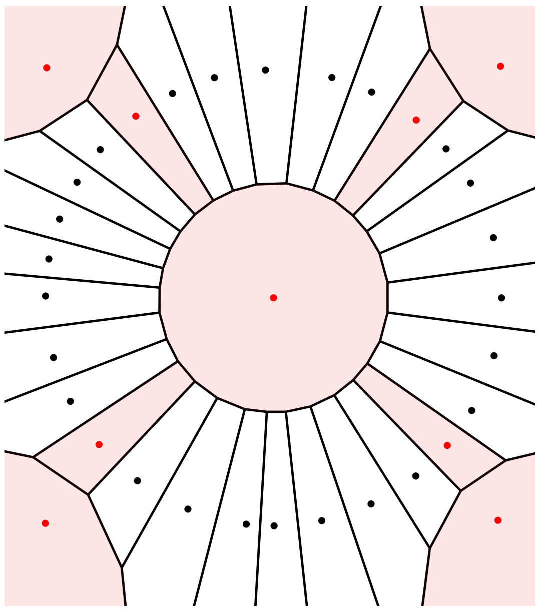


Figure 2 High complexity cells can occur in Voronoi diagrams. Such cells must be included in fringe verticies $Sample_j(k)$, illustrated in red for some point set T_j. This results in the complexity of the boundary of the interior sets $Cells(T_j, \overline{Sep\_j}^{\ell}(k))$, the connected components of white Voronoi cells, are of complexity $O(2^{4(j-i)})$.

One property of this sampling technique is that points in T_j with Voronoi cells in $Vor(T_j)$ of complexity at least k are included in T_i if $j > i$ and $j - i = O(\log k)$.

▶ **Lemma 2.1.** *If $p \notin T_i$ and $p \in T_j$, $i < j$, then the complexity of $Cell(T_j, p)$ is $O(2^{4(j-i)})$.*

The Jump function: definition. At the core of our nearest neighbor algorithm is the function *Jump*, defined as follows. We will find it helpful to use $NN_R(q)$ for a range $R = [l, r]$ to denote $NN(\cup_{i\in[l,r]} T_i, q)$.

Intuitively, a call to $Jump(i, j, q, p_i, e_i)$ is used when trying to find the nearest neighbor of q, and assuming we know the nearest neighbor of q in $T_1, T_2 \ldots T_{(i+j)/2}$ seeks to provide information on whether there are any points that could be the nearest neighbor of q in $T_{(i+j)/2+1} \ldots T_j$. This information could be either a simple *no*, or it could provide the nearest neighbor of q for some prefix of these sets. Additionally, the edge of an the Voronoi cell of the currently known nearest neighbor in the direction of the query point is always passed and returned to aide the search by limiting it to a single triangular piece of a Voronoi cell, the complexity of which we can bound in a way which does not hold for the full cell.

- **Input to** $Jump(i, j, q, p_i, e_i)$:
  - Integers i and j, where $j - i$ is required to be a power of 2. We use m to refer to $(j + i)/2$, the midpoint.
  - Query point q.
  - Point p_i where $p_i = NN(T_i, q)$.
  - The edge e_i on the boundary of $Cell(T_i, p_i)$ that the ray $\overrightarrow{p_i q}$ intersects.
- **Output**: Either one of two results, *Failure* or a triple $(j', p_{j'}, e_{j'})$
  - If *Failure*, this is a certificate that $NN_{(m,\min(j,f))}(q) \neq NN_{[1,\min(j,f)]}(q)$
  - If a triple $(j', p_{j'}, e_{j'})$ is returned, it has the following properties:

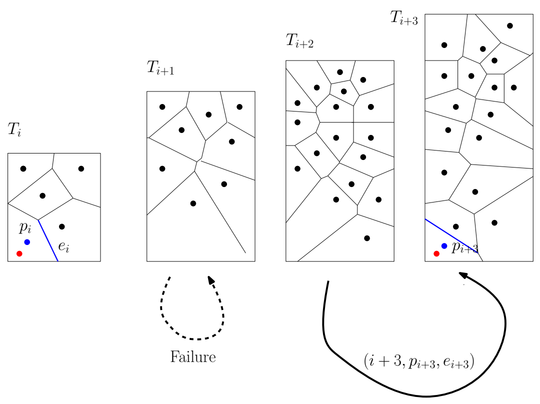


Figure 3 Two iterations of the Jump procedure. The query point q is shown in red. Points $p_i = NN(T_i, q)$, $p_{i+3} = NN(T_{i+3}, q)$, and edges e_i and e_{i+3} are shown in blue.

  * The integer j' is in the range $(m, j]$ and $NN_{(m,j')}(q) \neq NN_{[1,j')}(q)$.
  * The point $p_{j'}$ is $NN(T_{j'}, q)$.
  * The edge $e_{j'}$ is on the boundary of $Cell(T_{j'}, p_j)$ that the ray $\overrightarrow{p_{j'}q}$ intersects.

We show in the full version that *Jump* runs in $O(j - i)$ time.

The nearest neighbor procedure. A nearest neighbor query can be answered through a series of calls to the *Jump* function:

- Initialize $i = 1, j = 2$, p_1 to be $NN(T_1, q)$, and e_1 to be the edge of $Cell(T_1, p_1)$ crossed by the ray $\overrightarrow{p_1q}$; all of these can be found in constant time as $|T_1| = \Theta(1)$. Initialize $p_{nearest}$ to p_1.
- Repeat the following while $\frac{i+j}{2} \leq f$:
  - Run $Jump(i, j, q, p_i, e_i)$. If the result is failure:
    * Set $j = j + (j - i)$
  - Else a triple $(j', p_{j'}, e_{j'})$ is returned:
    * If $d(p_{j'}, q) < d(p_{nearest}, q)$ set $p_{nearest} = p_{j'}$
    * Set $i = j'$ and set $j = j' + 1$
- Return $p_{nearest}$

In the full version we provide the jump function's implementation, and via an amortized analysis show that we can answer a nearest neighbor query in $O(\log n)$ amortized time. Obtaining polylogarithmic insertion time is via the straightforward use of Bentely-Saxe rebuilding, and reducing is further to $O(\log^{1+\varepsilon})$ is described in the full version.

References

1 Pankaj K. Agarwal and Jirí Matousek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13(4):325–345, 1995. doi:10.1007/BF01293483.

2 Sarah R. Allen, Luis Barba, John Iacono, and Stefan Langerman. Incremental Voronoi diagrams. *Discrete and Computational Geometry*, 58(4):822–848, 2017. doi:10.1007/s00454-017-9943-2.

3 Jon Louis Bentley and James B. Saxe. Decomposable searching problems I: static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980. doi:10.1016/0196-6774(80)90015-2.

4 Timothy M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *Journal of the ACM*, 57(3):16:1–16:15, 2010. doi:10.1145/1706591.1706596.

5 Timothy M. Chan. Dynamic geometric data structures via shallow cuttings. *Discrete and Computational Geometry*, 64(4):1235–1252, 2020. doi:10.1007/s00454-020-00229-5.

6 Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. *Discrete and Computational Geometry*, 56(4):866–881, 2016. doi:10.1007/s00454-016-9784-4.

7 Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(2):133–162, 1986. doi:10.1007/BF01840440.

8 Bernard Chazelle and Ding Liu. Lower bounds for intersection searching and fractional cascading in higher dimension. *Journal of Computing & Systems Sciences*, 68(2):269–284, 2004. doi:10.1016/j.jcss.2003.07.003.

9 Kenneth L. Clarkson, Kurt Mehlhorn, and Raimund Seidel. Four results on randomized incremental constructions. *Computational Geometry*, 3:185–212, 1993. doi:10.1016/0925-7721(93)90009-U.

10 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008.

11 Olivier Devillers, Stefan Meiser, and Monique Teillaud. Fully dynamic Delaunay triangulation in logarithmic expected time per operation. *Computational Geometry*, 2:55–80, 1992. doi:10.1016/0925-7721(92)90025-N.

12 David P. Dobkin and Subhash Suri. Maintenance of geometric extrema. *Journal of the ACM*, 38(2):275–298, 1991. doi:10.1145/103516.103518.

13 Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, 1987. doi:10.1137/0216064.

14 Michael T. Goodrich. Planar separators and parallel polygon triangulation. *Journal of Computer and Systems Sciences*, 51(3):374–389, 1995. doi:10.1006/jcss.1995.1076.

15 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. *Discrete and Computational Geometry*, 64(3):838–904, 2020. doi:10.1007/s00454-020-00243-7.

16 Philip N. Klein, Shay Mozes, and Christian Sommer. Structured recursive separator decompositions for planar graphs in linear time. In *Symposium on Theory of Computing (STOC)*, pages 505–514, 2013. doi:10.1145/2488608.2488672.

17 Joseph S.B. Mitchell and Wolfgang Mulzer. Proximity algorithms. In Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 32, pages 849–874. CRC Press, 2017.

18 Ketan Mulmuley. *Computational geometry — an introduction through randomized algorithms*. Prentice Hall, 1994.

Flipping Plane Spanning Trees Compatibly [*]

Oswin Aichholzer, Joseph Dorfer, and Birgit Vogtenhuber

Institute of Algorithms and Theory, Graz University of Technology, Austria
`{oswin.aichholzer, joseph.dorfer, birgit.vogtenhuber}@tugraz.at`

Abstract

For compatible flip sequences between plane spanning trees on point sets in convex position we prove the happy edge property, show fixed-parameter tractability of the flip distance, and provide an upper bound of $\frac{5}{3}(n-1)$ for the flip distance. Additionally, we provide a framework to compare different happy edge properties.

1 Introduction

Let S be a finite point set in the plane in general position, that is, no three points lie on a common line. We call S a *convex* point set if the points in S no point in S lies in the interior of the convex hull of S. A plane straight-line graph of S is a graph with vertex set S and whose edges are straight line segments between pairs of points of S such that no two edges intersect except at a common endpoint. All graphs considered in this paper are straight-line graphs. For brevity we will omit the term straight-line.

Flip Graphs of Plane Spanning Trees. A *flip* between two plane spanning trees of S is the operation that removes an edge from a tree and adds another edge such that the resulting structure is again a plane spanning tree. We also denote this operation as an *unrestricted flip*. A restricted version of flips are so-called *compatible flips*, for which the removed edge and the added edge are not allowed to cross. The *(compatible) flip graph* of plane spanning trees of S has as its vertex set all such trees. Two vertices T_1, T_2 of this flip graph are connected with an edge if and only if T_1 can be transformed into T_2 by a single (compatible) flip. See Figure 1 for an example of flips in plane spanning trees .

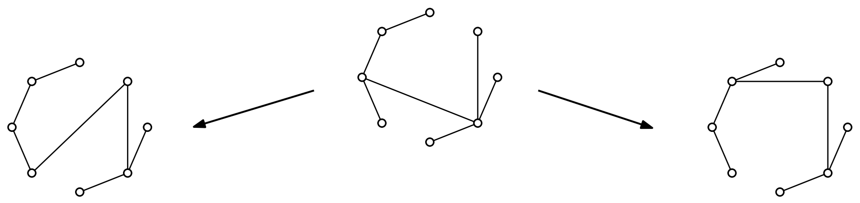


Figure 1 Two flips in a plane spanning tree starting from the tree in the middle. The flip to the left is not compatible since the added and removed added cross. The flip to the right is compatible.

Given an initial tree T_{in} and a target tree T_{tar}, a *(compatible) flip sequence* from T_{in} to T_{tar} is a path from T_{in} to T_{tar} in the (compatible) flip graph. The *(compatible) flip distance* between T_{in} and T_{tar} is the length of a shortest path between T_{in} and T_{tar} in the flip graph. Any (compatible) flip sequence of this length is called a *shortest (compatible) flip sequence*. See Figure 2 for an illustration of the concepts.

[*] This research was funded in part by the Austrian Science Fund (FWF) 10.55776/DOC183.

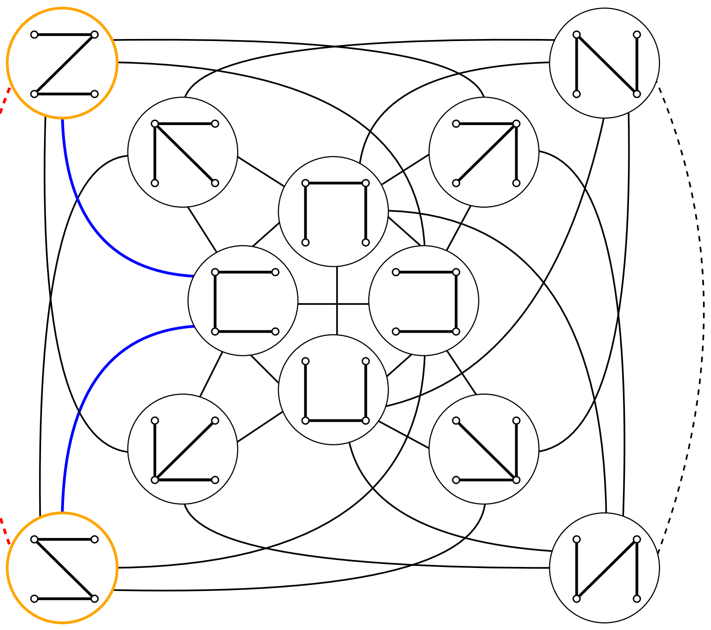


Figure 2 Flip graph of plane spanning trees on four vertices in convex position. Dashed edges denote flips that are part of the flip graph, but not the compatible flip graph. One pair of two plane spanning trees is marked in orange. Their shortest flip sequence of length one is marked in red (thicker dashed). A shortest compatible flip sequence of length two is marked in blue (thicker).

For any n-point set S, the flip graph of plane spanning trees on S is known to be connected and has radius exactly $n-2$ [3, 4]. Hence the diameter always lies between $n-2$ and $2n-4$. Since a lower bound for the diameter of $\lfloor \frac{3}{2}n \rfloor - 5$ for convex n-point sets in [9], the flip graph of plane spanning trees on convex point sets has received considerable attention. In the last few years, the upper bound for its diameter was improved to $2n - \Omega(log(n))$ in [1] and soon after to $2n - \Omega(\sqrt{n})$ in [7]. In the latter work, it was conjectured that the diameter is at most $\frac{3}{2}n$. The upper bound from [7] is constructive and, though not stated explicitly, only uses compatible flips. Hence it also provides an upper bound for the diameter of the according compatible flip graph. In [6], the diameter was bounded from above by cn with $c = \frac{1}{12}\left(22 + \sqrt{2}\right) \approx 1.95$, which marked the first linear improvement over the initial bound from [3]. Very recently, a lower bound of $\frac{14}{9}n - O(1)$ and an upper bound of $\frac{5}{3}n - 3$ on the diameter were achieved in [4], where the upper bound in general requires non-compatible flips.

Happy Edges. For any graph reconfiguration problem, *happy edges* are edges that lie in both the initial and target graph. A flip graph fulfills the *happy edge property* if there exists a shortest flip sequence between any two graphs that never flips happy edges. The happy edge property often is a good indication for the complexity of a graph reconfiguration problem. For example, for triangulations of simple polygons [2] and general point sets [11, 13], finding shortest flip sequences is computationally hard. The hardness proofs use counterexamples to the happy edge property as a key ingredient. Conversely, the happy edge property is known to hold for triangulations of convex polygons [14]. Though the complexity of finding shortest flip sequences is still open, the property yields multiple fixed-parameter tractable algorithms [5, 8, 12]. Moreover, the happy edge property holds for plane perfect matchings of convex point sets and shortest flip sequences can be found in polynomial time [10].

Contribution and Outline. In this work, we study different happy edge properties and the compatible flip graph of plane spanning trees on a convex point set.

Section 2 is dedicated to a more refined framework of happy edge properties that allows us to compare different graph reconfiguration problems. In Section 3, we prove that a happy edge property holds for compatible flips on plane spanning trees on convex point sets. From this happy edge property we derive a fixed-parameter tractable algorithm for finding shortest compatible flip sequences between pairs of spanning trees (Section 4), which moreover can be adapted for unrestricted flips in case they also fulfill the happy edge property. In Section 5, we show how to adapt strategies from [4] to obtain an upper bound of $\frac{5}{3}(n-1)$ for the diameter of the compatible flip graph of plane spanning trees on convex n-point sets.

2 Relations of Happy Edge Properties

In this section, we introduce a more refined distinction of happy edge properties.

▶ **Definition 2.1.** A graph reconfiguration problem where flips exchange edges fulfills the . . .
- . . . *(Weak) Happy Edge Property* if, from any initial graph G_{in} to any target graph G_{tar}, there exists a shortest flip sequence that does not flip happy edges.
- . . . *Strong Happy Edge Property* if, from any initial graph G_{in} to any target graph G_{tar}, any shortest flip sequence does not flip happy edges.
- . . . *Perfect Flip Property* if, whenever we can perform a flip f in G_{in} such that the resulting graph G_1 has one edge more in common with G_{tar} than G_{in} has (a.k.a. a *perfect flip*), then there exists a shortest flip sequence from G_{in} to G_{tar} that has G_1 as its first intermediate graph (that is, the flip sequence starts with f).

In a full version of the paper we prove the following relation to compare happy edge properties.

▶ **Proposition 2.2.** *Let P be a graph reconfiguration problem where flips exchange one edge.*
(i) If any flip sequence in P that flips at least one happy edge can be shortened by at least two flips, then P fulfills the Perfect Flip Property.
(ii) If for any flip in P from G to $(G \setminus \{e_1\}) \cup \{e_2\}$, e_1 and e_2 can never be in the same configuration, then the reverse direction in (i) holds.

We remark that all the introduced properties may or may not hold for certain graph reconfiguration problems. While the condition in Proposition 2.2(ii) is not fulfilled for trees, it does hold for example for triangulations.

3 Happy Edges in Plane Spanning Trees on Convex Sets

In [1], the authors formulated the *Weak Happy Edge Conjecture* for trees on convex point sets:

▶ **Conjecture 3.1** (Conjecture 17 in [1]). *For any two plane spanning trees T_{in} and T_{tar} on a convex point set, there is a shortest flip sequence from T_{in} to T_{tar} that does not flip happy edges.*

Based on an example from [1] for a different context, we first observe that the Perfect Flip Property holds neither for the unrestricted flip nor for the compatible flip on trees.

In contrast, we will show in this section that the Strong Happy Edge Property does hold for compatible flips on trees. Our first proof ingredient is the following lemma, which is an

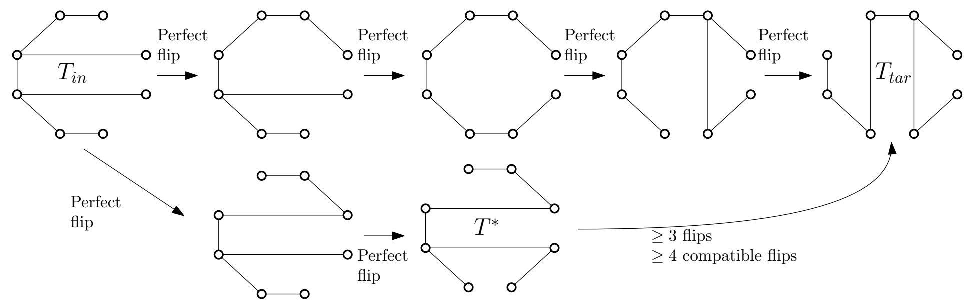


Figure 3 Counterexample to the Perfect Flip Property based on [1, Figure 7]. The top shows the shortest flip sequence. The bottom sequence starts with two perfect flips, but reaches a point where no more perfect flips are possible. Since there are two edges in the tree T^* that both cross two edges from the target tree, at least three additional flips or four additional compatible flips are needed.

extension of [1, Proposition 18] for compatible flips. Its proof can be found in a full version of our paper.

▶ **Lemma 3.2.** *Consider any point set S and any two plane spanning trees T_{in} and T_{tar} on S and any shortest compatible flip sequence from T_{in} to T_{tar}. If some edge e is removed and later added back, then some flip during that subsequence must add an edge f that crosses e.*

In [1], a *parking edge* is defined as an edge that appears in a flip sequence and that is not contained in $T_{in} \cup T_{tar}$. A second ingredient of our proof is Lemma 3.4, which verifies the compatible flip analogue (Lemma 3.4) of the following conjecture from [1].

▶ **Conjecture 3.3** (Conjecture 21 in [1]). *For any convex point set S and any two plane spanning trees T_{in} and T_{tar} on S, there is a shortest flip sequence from T_{in} to T_{tar} that only uses parking edges from the boundary of the convex hull of S.*

We Remark that in [1, Claim 22] it is shown that for unrestricted flips Conjecture 3.3 implies Conjecture 3.1.

▶ **Lemma 3.4.** *The analogue of Conjecture 3.3 for compatible flips holds.*

The proof of Lemma 3.4 can be found in the full version of the paper. Figure 4 shows the intuition behind that proof: One by one, we replace a parking edge f that is not on the boundary of the convex hull with a parking edge h on the convex hull boundary. To make this replacement possible, we change the order of the flips that happen while f is part of the tree. The edge f splits the convex point set into two sides, say A and B. Flips in either of the two sides can be executed independently from flips in the other side. Assume f is added when flipping from the tree T_{i_1-1} to T_{i_1} and removed in the flip from T_{i_2} to T_{i_2+1}. Exactly one of the sides, say side B, of T_{i_1-1} entirely contains a path that connects the two endpoints of f. The flips in the other side, say side A, can be executed before f gets added. Afterwards, we close a cycle in side A by adding the convex hull parking edge h and execute all the flips in side B. We conclude the new subsequence of flips by removing h and obtain the tree T_{i_2}.

We now show the Strong Happy Edge Property for compatible flips on trees.

▶ **Theorem 3.5.** *For any convex point set S and any two plane spanning trees T_{in} and T_{tar} on S, no shortest compatible flip sequence from T_{in} to T_{tar} removes and adds a happy edge. Any flip sequence that removes (and adds) a happy edge is at least one step longer.*

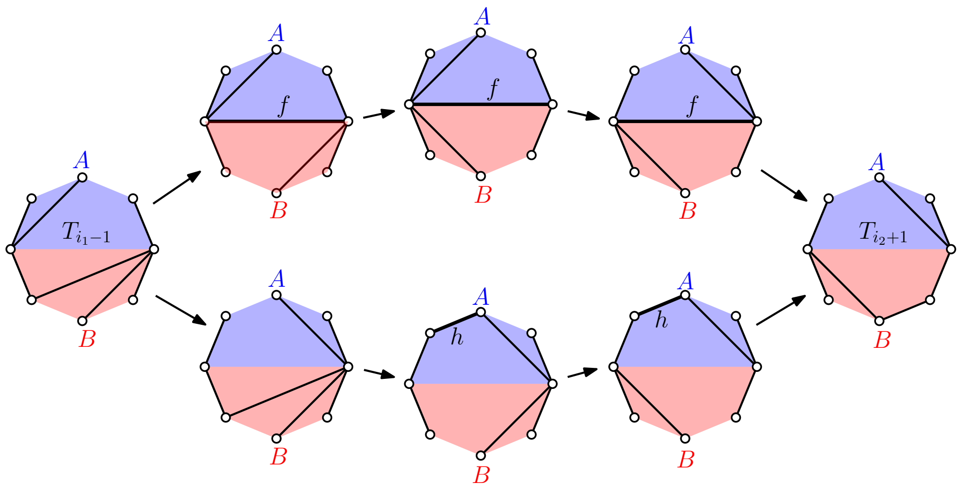


Figure 4 Reordering flips in Lemma 3.4. The top path is the part of the original flip sequence that contains f, the bottom path is the part of the reordered flip sequence that contains h.

Proof. Let $T_{in} = T_0, T_1,\ldots,T_k = T_{tar}$ be a compatible flip sequence that removes a happy edge e in a flip from T_i to T_{i+1}. By Lemma 3.4, there exists a shortest compatible flip sequence from $T_{i+1} = T'_{i+1},T'_{i+2},\ldots,T'_{tar} = T_{tar}$ that only uses parking edges from the convex hull boundary. Since e crosses neither and edge of $T_{in} \cap T_{tar}$ nor one on the convex hull boundary, e crosses no edge in the flip sequence $T_0,\ldots,T_i,T'_{i+1},\ldots,T_{tar}$. Hence, by Lemma 3.2, we can construct a shorter flip sequence that preserves e. ◀

By Proposition 2.2(i) and the example in Figure 3, Theorem 3.5 is best possible in the sense that there are flip sequences that flip a happy edge and cannot be shortened by two flips.

4 A Fixed-Parameter Tractable Algorithm

We next study the impact of Conjecture 3.1 on the complexity of finding the flip distance between two plane spanning trees on a convex point set. For that, we assume that Conjecture 3.1 is true, that is, shortest flip sequences preserve happy edges.

▶ **Theorem 4.1.** *If Conjecture 3.1 is true, then the flip distance k between two plane spanning trees on a convex point set is fixed-parameter tractable in k. This implication still holds if we only consider compatible flip sequences.*

The proof of Theorem 4.1 can be found in a full version of the paper. Here we show that the stronger assumption of Conjecture 3.3 implies fixed-parameter tractability.

Proving fixed-parameter tractability assuming Conjecture 3.3 holds. We divide the point set into components by cutting along happy edges and then find the shortest flip sequence for every remaining component individually, see Figure 5. Observe that all the happy edges in these components are boundary edges. If all edges in a component are happy, do not flip any edge in it. By Conjecture 3.3 there exists a shortest flip sequence that only uses edges from T_{in}, T_{tar} or the convex hull. Thus, in every step we need to choose one out of at most k unhappy edges to be removed. Further, there are at most $3k$ positions to add an edge, namely k from $T_{tar} \setminus T_{in}$ and $2k$ from gaps in the convex hull. Every component has one gap in the convex hull and gets an additional gap for every unhappy diagonal it contains. This yields a total of $(3k^2)^k$ flip sequences to check. ◀

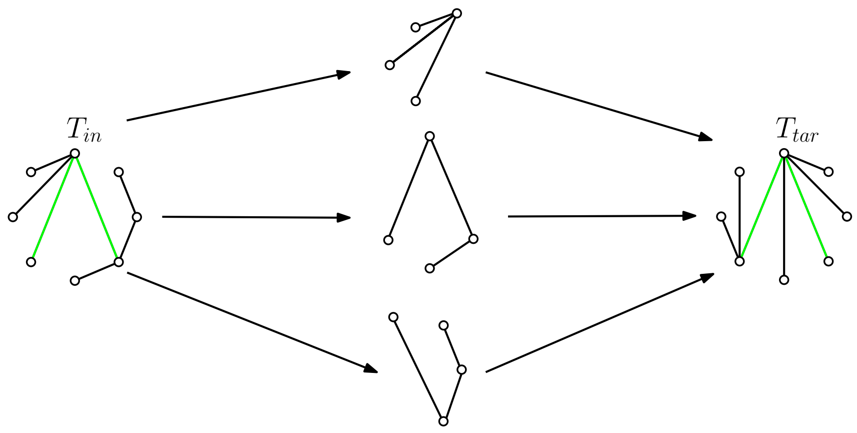


Figure 5 We split the initial tree T_{in} along its happy edges (colored in green) into three parts. From there we flip every part individually into its corresponding counterpart in T_{tar}.

▶ **Corollary 4.2.** *The compatible flip distance k_c between two plane spanning trees on a convex point set is fixed-parameter tractable in k_c.*

5 Improving the Upper Bound of the Compatible Flip Graph

By Lemma 3.4 there always exists a shortest compatible flip sequence that only flips edges from the symmetric difference $T_{in}\Delta T_{tar}$ and the convex hull. All the currently known improvements to the upper bound of the length of shortest flip sequences for unrestricted flips are built around this idea and use two different ways to flip edges: either invest two flips to flip an edge from $T_{in} \setminus T_{tar}$ to a convex hull edge and later flip that convex hull edge to an edge of $T_{tar} \setminus T_{in}$; or perform a single perfect flip from an edge of $T_{in} \setminus T_{tar}$ to an edge of $T_{tar} \setminus T_{in}$. Upper bounding the length of a flip sequence is then achieved by lower bounding the number of perfect flips.

In [4], edges of T_{in} are paired with edges of T_{tar} via a bijective mapping between the edges of each tree and $n-1$ convex hull edges. Roughly speaking, three sets of pairs of edges are identified such that $\approx \frac{n}{3}$ perfect flips can be guaranteed by the following procedure: Flip all edges except the ones from the largest set to the convex hull, flip all pairs of edges in the largest set perfectly, flip all other edges from the convex hull to their designated location.

Since one of the three described sets consists of pairs of edges that cross, the flip sequence resulting from this approach is not applicable for compatible flips. However, all other performed flips are compatible. In a full version of the paper we develop an approach how to regroup the pairs of crossing edges such that the resulting flip sequence is compatible while the length of the flip sequence increases by only one flip. This yields the following theorem.

▶ **Theorem 5.1.** *Between any two plane spanning trees T and T' on a convex point set with n vertices there exists a compatible flip sequence with at most $\frac{5}{3}(n-1)$ flips.*

A complete proof of Theorem 5.1 can be found in a full version of the paper.

6 Conclusion

We discussed a framework for different happy edge properties and showed relations between these properties. For the reconfiguration of plane spanning trees in convex point sets with compatible flips we showed that the happy edge property holds and that it implies fixed-parameter tractability of the flip distance. Further, we improved the upper bound for the compatible flip graph to match the upper bound of $\frac{5}{3}(n-1)$ that is known for the unrestricted flip. Some open problems related to our work are:

1. Can we close the gap between upper and lower bounds for the flip distance for both unrestricted and compatible flips? Do the flip distances differ or do they coincide?
2. Does the happy edge property still hold for flipping plane spanning trees if we drop one of the restrictions that either the point set is convex or the flips are all compatible?
3. What is the time complexity of finding shortest flip sequences for plane spanning trees?

Acknowledgements

We want to thank the anonymous referees for helpful remarks.

References

1 Oswin Aichholzer, Brad Ballinger, Therese Biedl, Mirela Damian, Erik D. Demaine, Matias Korman, Anna Lubiw, Jayson Lynch, Josef Tkadlec, and Yushi Uno. Reconfiguration of non-crossing spanning trees. *Journal of Computational Geometry*, 15:224–253, 2024. `doi:10.20382/jocg.v15i1a9`.

2 Oswin Aichholzer, Wolfgang Mulzer, and Alexander Pilz. Flip distance between triangulations of a simple polygon is NP-complete. *Discrete & Computational Geometry*, 54(2):368–389, June 2015. `doi:10.1007/s00454-015-9709-7`.

3 David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1):21–46, 1996. First International Colloquium on Graphs and Optimization. `doi:10.1016/0166-218X(95)00026-N`.

4 Håvard Bakke Bjerkevik, Linda Kleist, Torsten Ueckerdt, and Birgit Vogtenhuber. Flipping non-crossing spanning trees. *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2313–2325. `arXiv:2410.23809`, `doi:10.1137/1.9781611978322.77`.

5 Miguel Bosch-Calvo and Steven Kelk. An improved kernel for the flip distance problem on simple convex polygons. *Inf. Process. Lett.*, 182(C), 2023. `doi:10.1016/j.ipl.2023.106381`.

6 Nicolas Bousquet, Lucas de Meyer, Théo Pierron, and Alexandra Wesolek. Reconfiguration of Plane Trees in Convex Geometric Graphs. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:17, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.SoCG.2024.22`.

7 Nicolas Bousquet, Valentin Gledel, Jonathan Narboni, and Théo Pierron. A note on the flip distance between non-crossing spanning trees. *Computing in Geometry and Topology*, 2023. `doi:10.57717/cgt.v2i1.36`.

8 Sean Cleary and Katherine St. John. Rotation distance is fixed-parameter tractable. *Inf. Process. Lett.*, 109(16):918–922, 2009. `doi:10.1016/J.IPL.2009.04.023`.

9 Carmen Hernando, Ferran Hurtado, Alberto Márquez, Mercè Mora, and Marc Noy. Geometric tree graphs of points in convex position. *Discrete Applied Mathematics*, 93(1):51–66, 1999. doi:10.1016/S0166-218X(99)00006-2.

10 M. Carmen Hernando, Fabrizio Hurtado, and Marc Noy. Graphs of non-crossing perfect matchings. *Graphs and Combinatorics*, 18:517–532, 01 2002. doi:10.1007/s003730200038.

11 Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Computational Geometry*, 49:17–23, 2015. arXiv:1205.2425, doi:10.1016/j.comgeo.2014.11.001.

12 Joan Lucas. An improved kernel size for rotation distance in binary trees. *Information Processing Letters*, 110:481–484, 06 2010. doi:10.1016/j.ipl.2010.04.022.

13 Alexander Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 47(5):589–604, 2014. doi:10.1016/j.comgeo.2014.01.001.

14 Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *Journal of the American Mathematical Society*, 1(3):647–681, July 1988. doi:10.1090/S0894-0347-1988-0928904-4.

NumPSLA — An experimental research tool for pseudoline arrangements and order types

Günter Rote[1]

1 **Freie Universität Berlin, Institut für Informatik**
rote@inf.fu-berlin.de

Abstract

We present a program for enumerating all pseudoline arrangements with a small number of pseudolines and abstract order types of small point sets. This program supports computer experiments with these structures, and it complements the order-type database of Aichholzer, Aurenhammer, and Krasser. This system makes it practical to explore the abstract order types for 12 points, and the pseudoline arrangements of 11 pseudolines.

1 Introduction

Questions about finite configurations of points or lines are at the core of discrete geometry. As one example of an outstanding open question, we mention the rectilinear crossing number problem for the complete graph K_n: For a given set S of n points in the plane, draw all straight segments between points in S, and count the pairs of segments that cross. What is the smallest number that can be obtained?

The order type of a point set. This question and many other questions and algorithms in discrete and computational geometry depend only on the "combinatorial structure", which is typically captured by an orientation predicate: Consider a finite point set $S = \{p_1, \ldots, p_n\}$. For each triplet $p_i, p_j, p_k \in S$, we need to know whether they lie in clockwise or counterclockwise order, or whether they are collinear. This information is enough to determine, say, the number of convex hull vertices, or the crossing number.

The order-type database. It is useful if one can let the computer exhaustively check small examples. This may provide a sanity check for wild conjectures, or it may form the basis for quantitative results that hold in general. We will mention one example below. The prime tool that facilitates this approach is the order-type database of Aichholzer, Aurenhammer, and Krasser [1, 2] at Graz University of Technology from the early 2000's. Originally, it contained a point set (given explicitly by coordinates) for each of the 14,309,547 order types of 10 points, as well as for the smaller sets. These point sets are optimized to avoid degeneracies as much as possible. Later, the database was extended [4] to include the 2.3 billion order types of 11 points (see the second column of Table 1).

Over the years, the database has been enriched with useful information about each order type, ranging from the size of the convex hull to advanced characteristics that are hard to compute, such as the number of triangulations or the number of crossing-free Hamilton cycles. The database of order types with up to 10 points can be obtained from the website of the project[1], and it can be queried via an e-mail interface. The database for 11 points needs

[1] http://www.ist.tugraz.at/aichholzer/research/rp/triangulations/ordertypes/

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

102.7 GBytes (44 bytes per order type for two 16-bit coordinates per point). Obviously, the approach of storing a representative of every order type has currently reached its limits with 11 points. We take an alternative approach: *generating* order types from scratch.

Big results from small sets. We mention just one example of a result that rests on the order-type database. Aichholzer et al. [3, Theorem 1] proved that every set S of n points in general position contains $\Omega(n \log^{4/5} n)$ convex 5-holes, i.e., 5-tuples of points in convex position with no points of S in the interior. Harborth [11] showed in 1978 that every set of 10 points contains a convex 5-hole. From this, one gets an immediate lower bound of $\lfloor n/10 \rfloor$ 5-holes by partitioning S into groups of size 10 by vertical lines. Various improvements of the constant factor of this linear bound were obtained over the years. The superlinear bound $\Omega(n \log^{4/5} n)$ goes beyond what can be reached by this simple technique. Nevertheless, at the basis of its proof, there are some structural lemmas about sets of 11 points. These lemmas were checked with the help of a computer by exhaustive enumeration of order types.

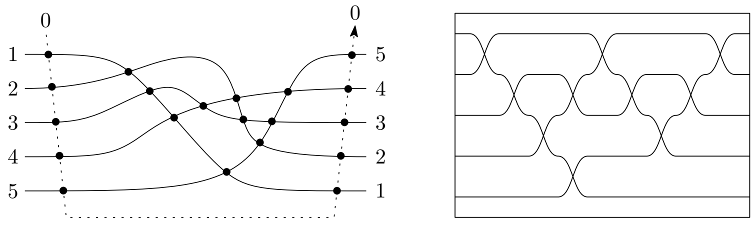
1.1 Line arrangements and pseudoline arrangements

The well-known duality

$$\text{point } (a,b) \longleftrightarrow \text{ line } y = ax - b \tag{1}$$

is a bijection between points and non-vertical lines. It swaps the role of points and lines, and it preserves incidences and above-below relationships. Thus, problems about points can be translated into problems about lines and vice versa.

Pseudoline arrangements and abstract order types of points. Pseudoline arrangements are a generalization of line arrangements. A pseudoline arrangement (PSLA) is a collection of unbounded curves, with the condition that any two curves intersect exactly once, and they cross at this intersection point. We refer to these curves as *pseudolines* or simply as *lines*. See Figure 1 for an example with 5 pseudolines. The middle and the right picture show a standard representation as a *wiring diagram*, in two different styles, as produced by our program. In a wiring diagram, the pseudolines run on n horizontal tracks, and they cross by swapping between adjacent tracks.



```
1 2-2-2-2 4-4-4-4 5
 X       X       X
2 1 3-3 4 2 3-3 5 4
   X   X   X   X
3-3 1 4 3-3 2 5 3-3
     X       X
4-4-4 1 5-5-5 2-2-2
       X
5-5-5-5 1-1-1-1-1-1
```

Figure 1 An arrangement of 5 pseudolines, extended by a pseudoline 0 "at infinity".

By duality, there is an analogous notion for point configurations, an *abstract order type* (AOT). We will elucidate this relation in Section 3. There are many other notions for these objects (rhombus tilings, oriented matroids of rank 3, signotopes), see [6, Chapter 6].

Our program focuses on pseudoline arrangements as the primary objects. The main reason is that they are easy to generate in an incremental way. Another reason is that they are easy to draw and to visualize.

Throughout this paper, we will assume general position. In other words, we restrict our attention to *simple* pseudoline arrangements, where no three lines go through a common point. In the setting of point sets, this corresponds to excluding collinear point triples.

	[A006247]	[A063666]	$\Delta =$	Δ	[A006245]
n	#AOT	#OT	#nonr. AOT	#AOT	#PSLA
3	1	1	0	0	2
4	2	2	0	0	8
5	3	3	0	0	62
6	16	16	0	0	908
7	135	135	0	0	24,698
8	3,315	3,315	0	0	1,232,944
9	158,830	158,817	13	0,01 %	112,018,190
10	14,320,182	14,309,547	10,635	0,07 %	18,410,581,880
11	2,343,203,071	2,334,512,907	8,690,164	0,37 %	5,449,192,389,984
12	691,470,685,682				2,894,710,651,370,536
13	366,477,801,792,538				2,752,596,959,306,389,652

Table 1 #AOT = number of abstract order types for n points. #OT = number of order types. #PSLA = number of (x-monotone) pseudoline arrangements with n pseudolines. These are the objects that the program actually enumerates one by one (almost, because we try to apply shortcuts). The column headings link to the Online Encyclopedia of Integer Sequences [15].

1.2 Overview

We will describe our algorithm for enumerating pseudoline arrangements, and we will apply it to enumerate abstract order types. None of the techniques that we use are novel, but we have tried to streamline and simplify the algorithms. In terms of speed, we can compete with the order type database, see [14, Appendix A] in the full version of this paper. The main distinction is, of course, that the order type database contains only *realizable* order types, and that they come with coordinates. For many applications, the restriction to realizable order types is not important, and coordinates are not needed. In these applications, our approach shows its strength. Mustering the 14 million 10-point abstract order-types takes 10–20 seconds. The 11-point sets can be handled in half an hour, and the 12-point sets take about 200 CPU hours. To this, one must of course add the time for whatever one wants to do with those order types. The program is trivially parallelizable, and with a powerful compute-cluster, it is feasible to go even for 13 points, see [14, Appendix E].

The program is available on GitHub [13]. It is written in C, using the CWEB system of structured documentation of Donald E. Knuth and Silvio Levy[2]. We have occasionally used the enumeration for research questions, and we hope that it finds other users.

2 Enumeration of pseudoline arrangements

We concentrate on x-monotone pseudoline arrangements, in which the curves are x-monotone. Every pseudoline arrangement can be drawn in an x-monotone way, but this incurs a choice: One of the unbounded faces must be selected as the *top face* T, and the opposite unbounded

[2] `http://tug.ctan.org/info/knuth/cwebman.pdf`

face will become the *bottom face* B. Then the lines run from left to right, and we number them from 1 to n as they appear from top to bottom on the left side. If they were straight lines, they would be numbered by increasing slope.

2.1 Representing a pseudoline arrangement

The vertices and edges of a pseudoline arrangement form a plane graph. Navigation in this graph and manipulation of it is greatly simplified by the fact that we have precise control over the vertices: There is a vertex for each pair of lines, and every vertex has degree 4. We thus store the edges in two 2-dimensional arrays *succ* and *pred* of successor and predecessor pointers. The entries $\mathit{succ}[j,k]$ and $\mathit{pred}[j,k]$ refer to the crossing between line k and the line j. We think of the lines as oriented from left to right. Then $\mathit{succ}[j,k]$ and $\mathit{pred}[j,k]$ point to the next and previous crossing on line j. For the reversed index pair $[k,j]$, we get the corresponding information for line k. Thus, in the example of Figure 1, $\mathit{succ}[2,3] = 5$, and accordingly, $\mathit{pred}[2,5] = 3$.

We can easily determine which of j and k enters the intersection (k,j) from the top and bottom: By our numbering convention, the line with the smaller index always enters above the other line, and to the right of the crossing, it lies below the other line.

The infinite rays on line j are represented by the additional line 0: $\mathit{succ}[j,0]$ is the first (leftmost) crossing on line j, and $\mathit{pred}[j,0]$ is the last crossing. The intersections on line 0 are cyclically ordered $1,..,n$. Thus, $\mathit{succ}[0,i] = i+1$ and $\mathit{succ}[0,n] = 1$.

2.2 Incremental generation of pseudoline arrangements

We generate a PSLA with n lines by inserting line n into a PSLA with $n-1$ lines, in all possible ways. Then each PSLA has a unique predecessor PSLA, and this imposes a tree structure on the PSLAs, see Figure 2. Our program explores this *enumeration tree* in depth-first order. If we number the children of each node in the order in which they are visited, this leads to a unique identifier for every node, and thus for every PSLA, analogous to the Dewey decimal classification that is used to classify books in libraries.

Inserting the n-th pseudoline into a PSLA of $n-1$ lines corresponds to threading a curve from the bottom face B to the top face T, see Figure 3. (We temporarily relax the requirement that the extra pseudoline has to be x-monotone.) Following Knuth [12, Section 9, p. 38], such a curve is called a *cutpath* [7]. This corresponds to a source-to-target path in the dual graph of the PSLA. Orienting the dual edges in the way how line n can cross them, namely, from below to above, leads to a directed acyclic graph (a DAG). We can enumerate all such paths in a backtracking manner. Since the DAG has no sinks other than the target vertex T, a path cannot get stuck, and thus the enumeration of the paths is simple and fast.

The whole algorithm is thus a double recursion. The outer recursion extends a PSLA by adding a pseudoline n. The inner recursion extends a partially drawn pseudoline n to the next crossing, see Figure 4. It is implemented by walking along the boundary of the face that has been entered through the last crossing. All upper edges of the face are candidate edges for the next crossing of line n, and we try them in succession. We have decided to walk in counterclockwise order around the face. This means that the paths for line n are generated in “lexicographic” order from right to left, as can be checked in Figure 2. [14, Appendix F] gives a self-contained PYTHON program that implements this enumeration algorithm.

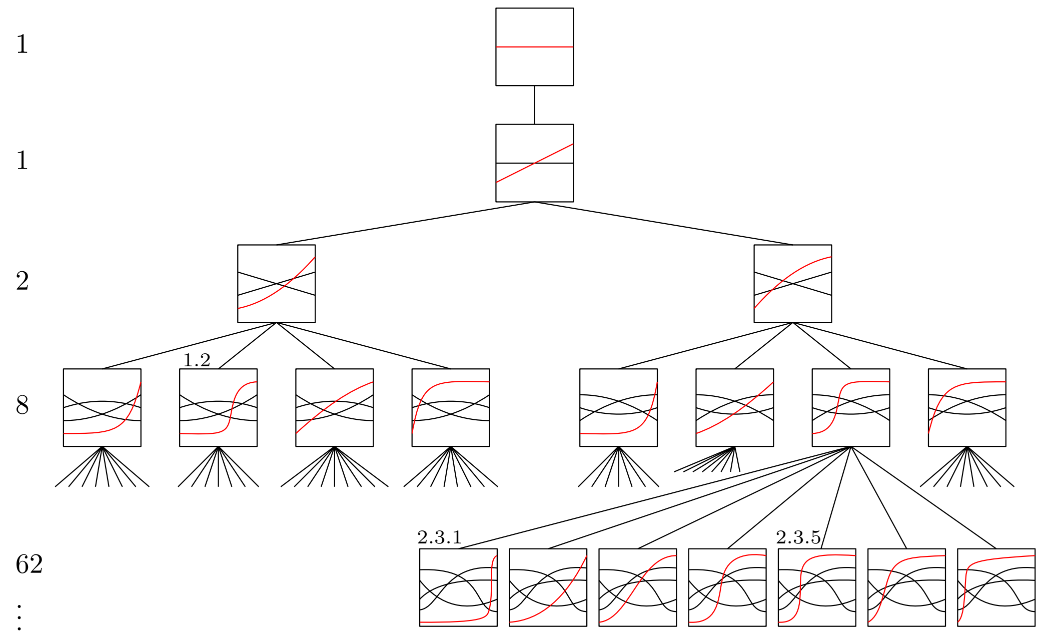


Figure 2 The first three three levels of the enumeration tree and a few nodes of the fourth level. The last inserted pseudoline is highlighted in red. For some nodes, the decimal notation is indicated.

3 Duality between pseudoline arrangements and abstract order types

The duality between pseudoline arrangements and abstract order types is not as straightforward as one would hope for. Figure 5 shows the intricate network of relationships. At the lower left corner, we find our favorite objects, the (x-monotone) PSLAs. The top right box refers to *oriented* abstract order types (OAOTs), where a point set is still distinguished from its reflection. (AOTs don't make this distinction.) The relations are discussed in [14, Appendix B]. The (x-monotone) PSLAs with n pseudolines correspond to OAOTs or AOTs with $n+1$ points, but the correspondence is not one-to-one. Different PSLAs may give rise to the same OAOT and AOT, and the algorithm has to take care of this ambiguity in order to enumerate OAOTs or AOTs without duplication. The details are given in [14, Appendix D].

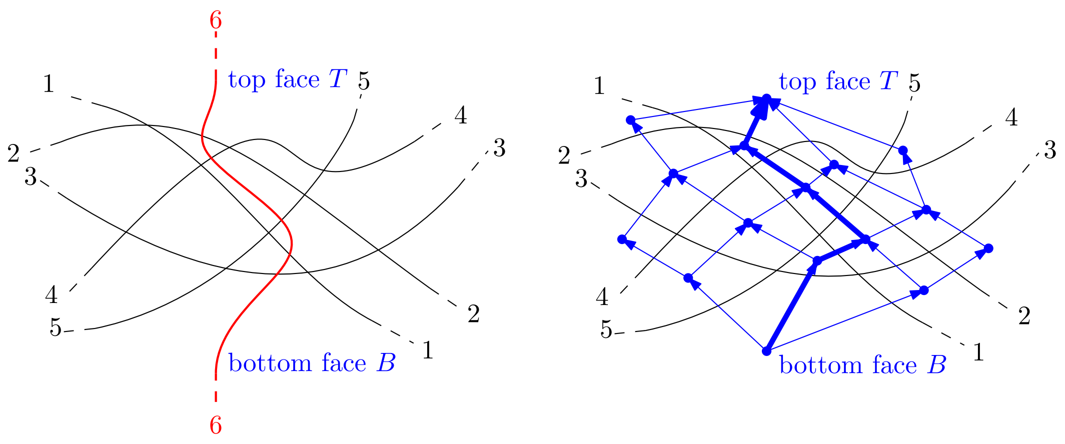


Figure 3 Left: Threading line 6 through a PSLA of 5 lines. Right: The dual DAG of this PSLA

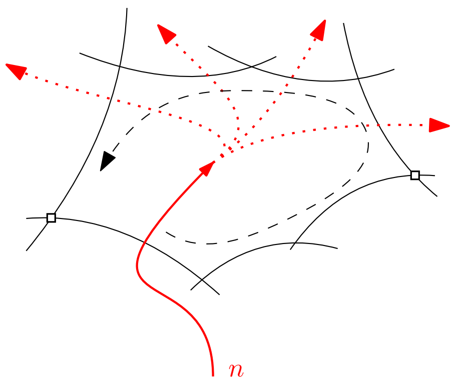


Figure 4 Continuing line n after entering a face.

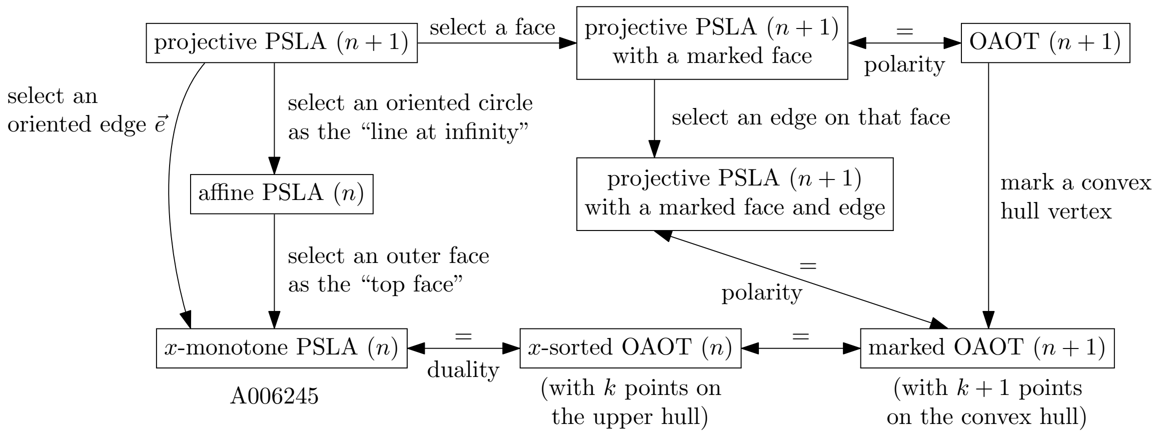

projective PSLA $(n+1)$
select a face
projective PSLA $(n+1)$ with a marked face
$=$ polarity
OAOT $(n+1)$
select an oriented edge $\vec{e}$
select an oriented circle as the "line at infinity"
select an edge on that face
affine PSLA (n)
projective PSLA $(n+1)$ with a marked face and edge
mark a convex hull vertex
select an outer face as the "top face"
$=$ polarity
x-monotone PSLA (n)
$=$ duality
x-sorted OAOT (n)
$=$
marked OAOT $(n+1)$
A006245
(with k points on the upper hull)
(with $k+1$ points on the convex hull)

Figure 5 Relation between different concepts. A unidirected arrow indicates a specialization.

4 Parallelization

We have implemented a trivial way to parallelize the enumeration. The user can choose a *split level*, usually 8. The program will then work normally up to level 8 of the tree, that is, it will enumerate all 1,232,944 PSLAs with 8 lines, but it will only expand a selection of these PSLAs. The selection is determined as follows. As the PSLAs with 8 lines are enumerated, a running counter is incremented, thus assigning a number between 1 and 1,232,944 to each PSLA. We specify a modulus m and a value k. Then the program will expand only those nodes whose number is congruent to k modulo m. By running the program for $k = 1, \dots, m$, the work is split into m roughly equal parts.

5 Enumerating only the realizable AOTs

We implemented a provision to enumerate only the (realizable) order types of points sets, for up to 11 points, to make the results comparable with those of the order-type database: There is an option to specify an *exclude-file* for the program. The exclude-file is a sorted list of decimal codes for tree nodes that should be skipped.[3] The exclude-files were prepared with the help of the order-type database. Essentially, we are storing the AOTs that are *not* realizable, which is a tiny minority compared to the realizable ones, see Table 1. Still, the exclude-file for up to 11 points has 8,699,559 entries and needs 184.6 MBytes. (With some technical effort, like eliminating common prefixes or a compressed binary format, one could reduce this space requirement significantly.)

6 Experiments and Extensions

We have gathered some statistics about various quantities for PSLAs and AOTs, such as the number of cutpaths, the number of hull vertices, the number of halving-lines, or crossing numbers. The results are reported in [14, Appendix E].

There are many ways in which one could think of extending the program.

1. We have concentrated on AOTs. PSLAs were used only as a tool to enumerate AOTs, but PSLAs could also be considered in their own right. They might be counted or classified with respect to different criteria, like projective equivalence classes or affine equivalence classes (cf. Figure 5).
2. "Partial" pseudoline arrangements, in which lines are not forced to cross; see Figure 6.

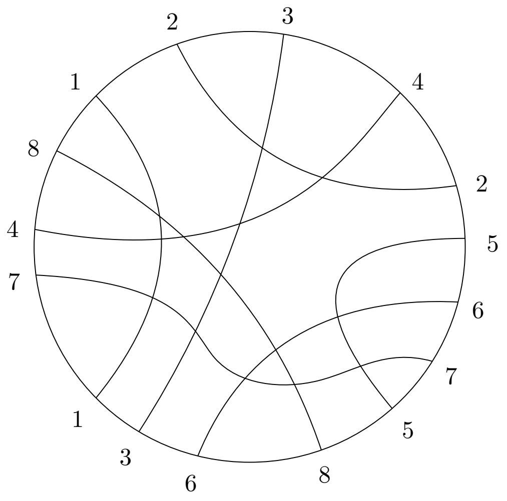


Figure 6 Example of a partial PSLA

3. Non-simple pseudoline arrangements, in which more than two pseudolines are allowed to cross in a point. In the language of oriented matroids, they correspond to nonuniform oriented matroids, and they have be enumerated by a method of Finschi and Fukuda [9], also in higher dimensions, see [8] for a catalog. These are much more numerous, see also Table 1 in [10], where also the realizability is considered. Handling them by our approach would involve a redesign of the data structures from scratch.
4. Random generation. It is easy to generate a random PSLA by diving into the tree randomly. This random selection will, however, be far from uniform, see [14, Appendix E.2].
5. A side issue are nice drawings of pseudoline arrangements. The wiring diagram is simple to obtain but it is very jagged. Stretchability can be a very hard problem. Constructing

[3] Currently the exclude-file feature does not work together with the parallelization feature. (For 11 points, the program should anyway be fast enough without parallelization.)

a drawing in which the pseudolines don't "bend too much" would be an interesting challenge. (Maybe it would be an idea for a Geometric Optimization Challenge[4], perhaps in connection with the random generation method mentioned above.)

Acknowledgements. We thank the High-Performance-Computing Service of FUB-IT, Freie Universität Berlin [5] for computing time.

References

1 O. Aichholzer, F. Aurenhammer, and H. Krasser. Enumerating order types for small point sets with applications. In *Proc.* 17^{th} *Ann. ACM Symp. Computational Geometry*, pages 11–18, Medford, Massachusetts, USA, 2001. doi:10.1145/378583.378596.

2 Oswin Aichholzer, Franz Aurenhammer, and Hannes Krasser. Enumerating order types for small point sets with applications. *Order*, 19(3):265–281, 2002. doi:10.1023/A:1021231927255.

3 Oswin Aichholzer, Martin Balko, Thomas Hackl, Jan Kynčl, Irene Parada, Manfred Scheucher, Pavel Valtr, and Birgit Vogtenhuber. A superlinear lower bound on the number of 5-holes. *Journal of Combinatorial Theory, Series A*, 173:105236, 2020. doi:https://doi.org/10.1016/j.jcta.2020.105236.

4 Oswin Aichholzer and Hannes Krasser. Abstract order type extension and new results on the rectilinear crossing number. *Computational Geometry*, 36(1):2–15, 2007. Special Issue on the 21st European Workshop on Computational Geometry. doi:10.1016/j.comgeo.2005.07.005.

5 Loris Bennett, Bernd Melchers, and Boris Proppe. Curta: A general-purpose high-performance computer at ZEDAT, Freie Universität Berlin, 2020. doi:10.17169/refubium-26754.

6 Stefan Felsner. *Geometric Graphs and Arrangements.* Advanced Lectures in Mathematics. Vieweg, Wiesbaden, 2004. URL: http://page.math.tu-berlin.de/~felsner/Buch/gga-book.pdf, doi:10.1007/978-3-322-80303-0.

7 Stefan Felsner and Pavel Valtr. Coding and counting arrangements of pseudolines. *Discrete & Comput. Geom.*, 46:405–416, 2011. doi:10.1007/s00454-011-9366-4.

8 Lukas Finschi. Homepage of oriented matroids. https://finschi.com/math/om/. Accessed 2025-02-25.

9 Lukas Finschi and Komei Fukuda. Generation of oriented matroids - A graph theoretical approach. *Discret. Comput. Geom.*, 27(1):117–136, 2002. doi:10.1007/S00454-001-0056-5.

10 Komei Fukuda, Hiroyuki Miyata, and Sonoko Moriyama. Complete enumeration of small realizable oriented matroids. *Discret. Comput. Geom.*, 49(2):359–381, 2013. doi:10.1007/S00454-012-9470-0.

11 Heiko Harborth. Konvexe Fünfecke in ebenen Punktmengen. *Elemente der Mathematik*, 33:116–118, 1978. URL: http://eudml.org/doc/141217.

12 Donald E. Knuth. *Axioms and Hulls*, volume 606 of *Lecture Notes in Computer Science.* Springer-Verlag, Heidelberg, 1992. doi:10.1007/3-540-55611-7.

13 Günter Rote. NumPSLA — an experimental research tool for pseudoline arrangements and order types. GitHub repository, 2024. URL: https://github.com/guenterrote/NumPSLA.

14 Günter Rote. NumPSLA — an experimental research tool for pseudoline arrangements and order types, 2025. arXiv:2503.02336.

15 The One-Line Encyclopedia of Integer Sequences. URL: http://oeis.org/.

[4] https://cgshop.ibr.cs.tu-bs.de/

Dihedral f-Tilings of the Sphere Induced by the Möbius Triangle $(2, 3, 4)$

Catarina Avelino*[1], Hoi Ping Luk†[2], and Altino Santos*[3]

1 Centre of Mathematics of the University of Minho - UTAD Pole, University of Trás-os-Montes and Alto Douro, Portugal.
cavelino@utad.pt

2 Západočeská univerzita v Plzni, Česká republika.
hoi@connect.ust.hk

3 Centre of Mathematics of the University of Minho - UTAD Pole, University of Trás-os-Montes and Alto Douro, Portugal.
afolgado@utad.pt

Abstract

We classify the special families of dihedral folding tilings of the sphere induced by the Möbius triangle $(2, 3, 4)$. Tilings in general are challenging to enumerate. The ones in our study emerge from the study of isometric foldings in the sphere and meet at a juncture of the triangle group $\Delta(2, 3, 4)$ and monohedral tilings of the sphere. The juxtaposition enables us to overcome the challenges in enumerating the tilings as a constraint satisfaction problem and unify the related classifications.

Related Version arXiv:2411.05973

1 Introduction

Tilings have been continuously studied for the long-held fascination with them. Milestones include the classification of the Wallpaper groups [15, 23], the solution to Hilbert's 18th problem [10, 19, 24], the discovery of Penrose tilings [25], the classification of the isohedral tilings of the plane [17], and most recently the discovery of aperiodic monotiles [29] of the plane as well as the classification of edge-to-edge monohedral tilings of the sphere [8, 9, 13, 14, 18, 20, 30, 32, 33].

The essence of many tiling problems concerns the existence of tilings under a set of constraints. The existence can be formulated as the solutions to the corresponding constraint satisfaction problems. The constraints are defined by the chosen types of tiles and how the tiles can be put together. In general, these constraint satisfaction problems are however not easy to solve merely by brute force. For example, despite the compactness of the sphere $\mathbb{S}^2$ and a concrete choice of tiles, without strong assumptions (such as global symmetry), the construction of a tiling can only start at a single tile and goes from tile to tile. The recurrence is further complicated by the multiple ways of arranging incident tiles at a vertex and the available choices of tiles. This is why the heavy machinery in [13, 14] is in vain in front of certain problems. Ours is one of them. The problem is a part of the programme to classify spherical tilings of folding type and the concrete choices of tiles emerge from a natural progression. Unlike the previous effort, enumeration in this one would be challenging

* This research was partially financed by Portuguese Funds through FCT (Fundação para a Ciência e a Tecnologia) within the projects UID/00013 of CMAT-UTAD, Center of Mathematics of University of Minho, and UID/04621 of CEMAT/IST-ID, Center for Computational and Stochastic Mathematics, Instituto Superior Técnico, University of Lisbon.

† The research was supported in part by the funding of Academic Career in Pilsen 2024 under Plzeňský kraj a Západočeská univerzita v Plzni.

by recursion, let alone by hand. By identifying a choice of tiles as the Möbius triangle $(2, 3, 4)$, the problem is placed under a unifying framework, as the hardest case in a three-case problem. It then reveals that all three cases originate from two mother tilings. This not only grants us clarity on the complexity but also unifies the current work in the classification.

The subject of this paper can be traced back to the juxtaposition of two topics. One centres on the function spaces of isometric foldings between Riemannian manifolds. An f-tiling is the set of singularities of an isometric folding $g : \mathbb{S}^2 \to S$ (definition in [26]) where S is a smooth oriented Riemannian surface. Foldings help to explain structural stability in catastrophes (catastrophe theory) [27]. It has been conjectured for $S = \mathbb{S}^2$ that any isometric folding is homotopic, through isometric foldings, to the trivial isomeric folding $f(x, y, z) := (x, y, |z|)$ or to f composed with the reflection in the xy-plane [26, Example 4]. The classification of f-tilings provides ample testing cases for the conjecture, which were previously unavailable. Another topic is geometric group theory where the triangle groups are studied as realisations via reflections across the edges of Möbius triangles. In general, Möbius triangles are Schwarz triangles $\Delta(l, m, n)$ with $l, m, n \in \mathbb{N}$. They are related to the solutions of hypergeometric functions [21] and the symmetry study of graphs and maps [28].

1.1 The Problem

The spherical tilings we study are *dihedral*, meaning that in each tiling some tiles are congruent to one polygon while the others are congruent to a different polygon. The polygons are called the *prototiles* of a tiling. If a tiling has exactly one prototile, then it is called *monohedral*. A tiling is called *edge-to-edge* if no vertex lies in the interior of an edge.

We focus on spherical dihedral tilings of *folding type* (or *f-tilings* for short), meaning that 1) they are edge-to-edge, and 2) at each vertex, its degree is even and ≥ 4, and the sums of alternate angles are π. The sums of alternate angles at a vertex of even degree means that for the $2k$ (for some integer $k \geq 2$) angles $\alpha_1, \alpha_2, ..., \alpha_{2k}$ labeled cyclically,

$$\sum_{i=1}^{k} \alpha_{2i-1} = \sum_{i=1}^{k} \alpha_{2i} = \pi. \tag{1}$$

The degree of a vertex is necessarily even if (1) is satisfied. Note that these two conditions on a vertex are consequences of the Hopf degree of an isometric folding f (necessarily continuous) and $\mathbb{S}^2$ being closed and orientable [26, Theorem 5]. The even degree condition implies a triangle in the tiling, explained by the following lemma.

▶ **Lemma 1.1.** *A triangle-free edge-to-edge spherical tiling has a vertex of degree* 3.

Proof. In the underlying graph of such a tiling, let f_m, v_k denote the numbers of m-gons for $m \geq 4$ and vertices of degree k for $k \geq 3$, and let v, e, f denote the total numbers of vertices, edges and faces, respectively. Then $f = \sum_{m\geq 4} f_m$ and $v = \sum_{k\geq 3} v_k$. Recall the Euler's polyhedral formula and the Dehn-Sommerville formulae [22],

$$v - e + f = 2, \tag{2}$$

$$2e = \sum_{k\geq 3} kv_k, \tag{3}$$

$$2e = \sum_{m\geq 4} mf_m. \tag{4}$$

For $m \geq 4$, the last formula implies $2e \geq 4f$. Combining the first two formulae gives

$$2 = v - e + f \leq v - e + \tfrac{e}{2} = v - \tfrac{e}{2} = \sum_{k\geq 3} \left(1 - \tfrac{k}{4}\right) v_k,$$

which implies $v_3 > 0$. Namely there is a degree 3 vertex. ◀

In view of the realisation of the triangle group $\Delta(2,3,4)$, we explain below how the prototiles in this paper can be understood under a general framework. As one of the reflection groups of the sphere, $\Delta(2,3,4)$ is represented as follows,

$$\Delta(2,3,4) = \langle u, v, w \,|\, u^2 = v^2 = w^2 = (uv)^2 = (vw)^3 = (wu)^4 \rangle,$$

where u, v, w are the three generators. The group is the automorphism group of a spherical tiling by the spherical triangle with angles $\frac{1}{2}\pi$, $\frac{1}{3}\pi$, $\frac{1}{4}\pi$, which is known as the Möbius triangle $(2,3,4)$. The angles $\alpha = \frac{1}{4}\pi, \beta = \frac{1}{3}\pi, \gamma = \frac{1}{2}\pi$ (first picture of Figure 1) means that the opposite edges have distinct lengths a, b, c, pictorially represented by $\|$, | and | respectively.

The Möbius triangle is one of the two prototiles in each case. The other prototile is *induced* by the Möbius triangle under the reflections of the triangle group $\Delta(2,3,4)$. The *induced prototile* is one of the three cases given by gluing two mirror images of the Möbius triangle along an edge: a kite (the second picture) along the c-edge, an isosceles triangle (the third picture) along the b-edge, or an isosceles triangle (the fourth picture) along the a-edge. The angles of the kite are denoted by $\alpha^2, \beta^2, \gamma, \gamma$, where α^2 (resp. β^2) denotes 2 copies of α (resp. β). The angle notations in the isosceles triangles are defined similarly. Let $\bar{x} = 2x$ for $x = a, b$. Then the prototiles are referred to as *the Möbius triangle*, *the kite* and *the isosceles triangles* $\triangle \bar{a}c^2, \triangle \bar{b}c^2$. The dihedral f-tilings with the Möbius triangle and one of the three induced prototiles are referred to as the dihedral f-tilings *induced by* the Möbius triangle.

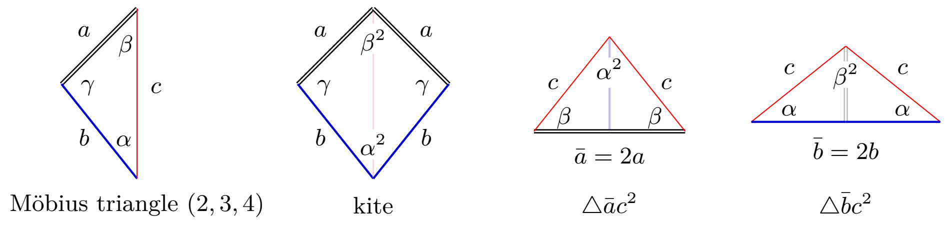


Figure 1 The Möbius triangle $\triangle abc$, the kite $\square a^2b^2$, and the isosceles triangles $\triangle \bar{a}c^2$ and $\triangle \bar{b}c^2$, $\alpha = \frac{1}{4}\pi$, $\beta = \frac{1}{3}\pi$ and $\gamma = \frac{1}{2}\pi$

The complexity in enumerating the tilings having the kite is beyond manual effort, which has led to the above framework. In general, the framework unifies significant classifications of dihedral f-tilings under Möbius triangles. The tiling having automorphism group $\Delta(2,3,4)$ is one of the two mother tilings. Both mother tilings in plane drawings are shown in Figure 2.

We state below the problem and the version in terms of a constraint satisfaction problem.

▶ **Problem.** Determine the dihedral f-tilings induced by the Möbius triangle $(2,3,4)$.

▶ **Problem** (The Related Constraint Satisfaction Problem)**.** Subject to the constraints of the chosen prototiles together with each vertex having an even degree and its sums of alternate angles equal to π, enumerate the unique tilings up to isometry.

2 Main Result

▶ **Theorem 2.1.** *There are a total of* 123 *dihedral f-tilings induced by the Möbius triangle* $(2,3,4)$*. Among them, there are*

1. 104 *dihedral f-tilings by the Möbius triangles and the kites; and*
2. 12 *dihedral f-tilings by the Möbius triangles and the* $\triangle \bar{a}c^2$*-triangles; and*
3. 7 *dihedral f-tilings by the Möbius triangles and the* $\triangle \bar{b}c^2$*-triangles.*

The last two cases were previously completed in [4] and [7] respectively.

Sketch of the Proof. Reversing the gluing in the second prototile results in two identical Möbius triangles. Then in a dihedral tiling, dividing each tile congruent to the second prototile into the Möbius triangles results in a monohedral tiling. The classification theorem in [13, 30] states that such a tiling is either the barycentric subdivision BO of the octahedron O and its flip modification FBO (plane drawings in Figure 2).

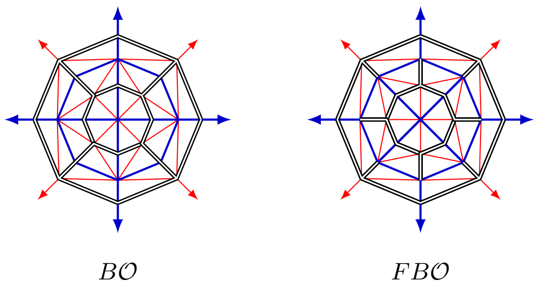


Figure 2 The plane drawings of the barycentric subdivision of the octahedron BO and its flip modification FBO; the arrows in each picture converge to a single vertex

In view of the "universality" of the mother tilings BO, FBO, the dihedral f-tilings are determined by the presence (or the absence) of each division edge x in one of BO, FBO, subject to the folding conditions for a fixed $x = a, b$ or c. We call a result from this process an *x-edge assignment* (in BO or FBO) or simply an *edge assignment*. For example, when $x = c$, the dashed lines in BO and FBO in Figure 3 indicate the locations where c-edges is assigned or otherwise. In BO, the labels, T_i for $i \in I := \{1, ..., 8\}$ and S_j for $j \in J := \{1, ..., 6\}$, represent the vertices corresponding to the cube (dual to the octahedron) and the octahedron, respectively. In FBO, the vertices after a $\frac{1}{4}\pi$-rotation in the inner hemisphere of BO are denoted as T'_1, T'_2, T'_3, T'_4 and S'_2, S'_3, S'_4, S'_5.

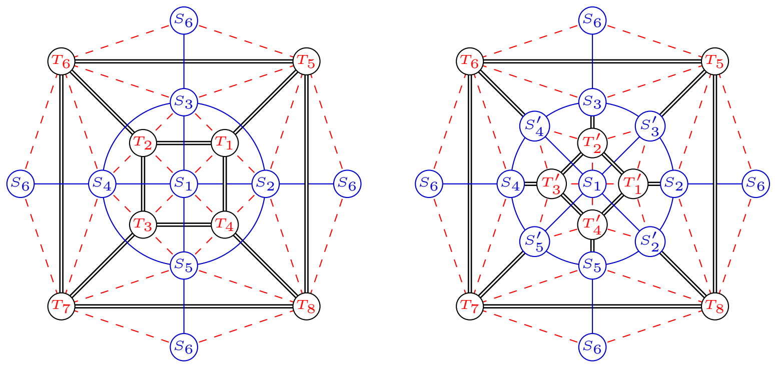


Figure 3 Locations of c-assignments in BO and FBO marked by dashed lines

Since the dihedral f-tilings are obtained by x-edge assignments in BO and FBO for fixed $x = a, b$ or c. It suffices to determine which assignments are unique up to isometry. For that, we use the geometric models (Figure 4) having equivalent position vectors of the vertices in $\mathbb{R}^3$ and then use their corresponding automorphism group to check for isometry between assignments. The models for $B\mathcal{O}$ and $FB\mathcal{O}$ are the spherical deltoidal icositetrahedron $o\mathcal{C}$ (Conway's notation) and the spherical pseudo-deltoidal icositetrahedron $Fo\mathcal{C}$ respectively. From Figure 4, it can be seen that the two are related via rotating the lower hemisphere (corr. to the inner hemisphere in Figure 3) along the equator. The automorphism group of $o\mathcal{C}$ is the triangle group $G = \Delta(2,3,4)$ (the octahedral symmetry), whereas the automorphism group G' of $Fo\mathcal{C}$ follows from the orbit-stabiliser theorem [16, Theorem 16.16]. ◀

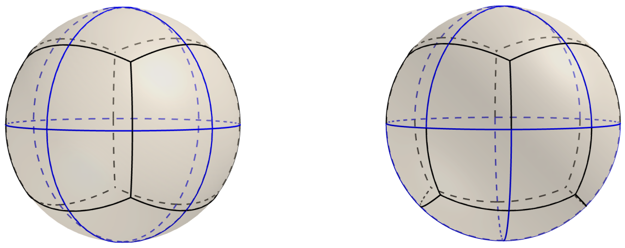


Figure 4 Models used for the c-edge assignment – the spherical deltoidal icositetrahedron $o\mathcal{C}$ and the spherical pseudo-deltoidal icositetrahedron $Fo\mathcal{C}$

3 The Algorithm

Pseudocode in Algorithm 1 outlines the algorithm for the c-edge assignment based on Figure 3 where the dashed lines mark where a c-edge is to be assigned or not. The degree assignment procedure counts how many c-edges are to be assigned to a vertex T_i. Since the existing degree of T_i is 3, to have an eventual even degree, only 1 or 3 is assigned to T_i. After filtering out the duplicates, c-edges are assigned based on the non-isomorphic degree assignments. Finally, duplicates in c-edge assignment are filtered out. The other cases are done similarly.

Algorithm 1 Degree & Edge Assignments for the *c*-Edge Assignment

```
procedure ASSIGN SUITABLE DEGREE TO EACH $T_i$ FOR $i = 1, ..., 8$
    for each $T_i$ do
        Deg_Assignment: assign suitable degree to $T_i$ so that $T_i$ is even
        keep Deg_Assignment
        if Deg_Assignment_List is empty then
            add Deg_Assignment to Deg_Assignment_List
        else
            for $\sigma \in \mathcal{G}$ do
                if $\sigma$Deg_Assignment $\in$ Deg_Assignment_List then
                    Is_isomorphic: true
            if Is_isomorphic = true then
                add Deg_Assignment to Deg_Assignment_List
procedure ASSIGN EDGE(S) TO EACH $T_i$ FOR $i = 1, ..., 8$ AND NEIGHBOURING $S_j$ FOR
$j = 1, ..., 6$ W.R.T. ASSIGNED DEGREE
    for each Deg_Assignment do
        Edge_Assigment: assign edge(s) adjacent $T_i, S_j$
        if Edge_Assigment satisfy folding conditions then
            keep Edge_Assigment
    if Edge_Assignment_List is empty then
        add Edge_Assignment to Edge_Assignment_List
    else
        for $\sigma \in \mathcal{G}$ do
            if $\sigma$Edge_Assignment $\in$ Edge_Assignment_List then
                Is_isomorphic: true
        if Is_isomorphic = true then
            add Edge_Assignment to Edge_Assignment_List
```

4 Conclusion

We conclude with the plane drawings of the *c*-edge assignment (unique up to isometry) and the link to all the tilings in 3D, `https://www.geogebra.org/m/zfnap4pe`.

Acknowledgments. The authors express their sincere gratitude for all the helpful comments and constructive criticisms from the reviewers. The second author also thanks the organisers for their assistance with all the submission-related matters.

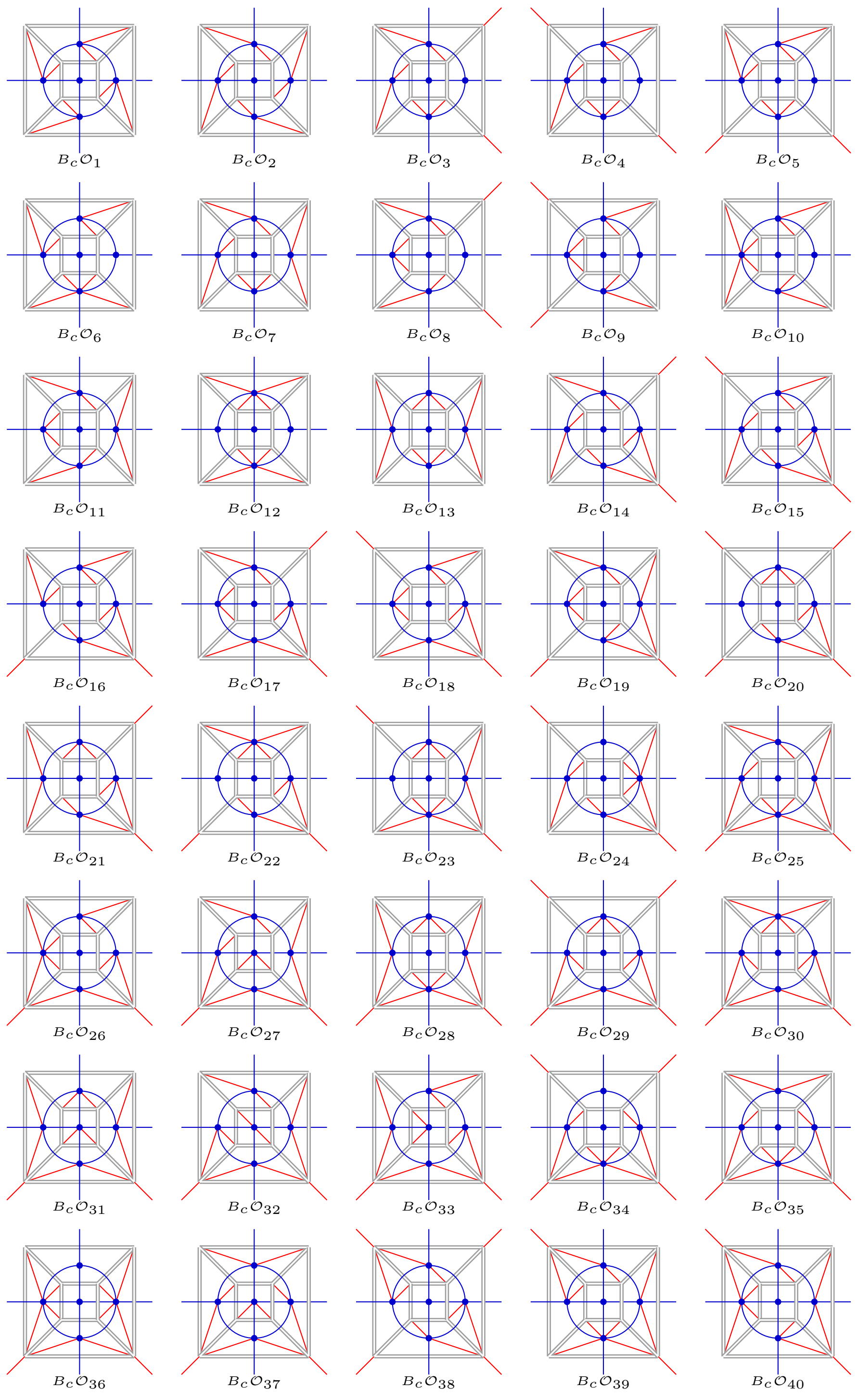


Figure 5 The c-edge assignments in $B\mathcal{O}$ up to isomorphism

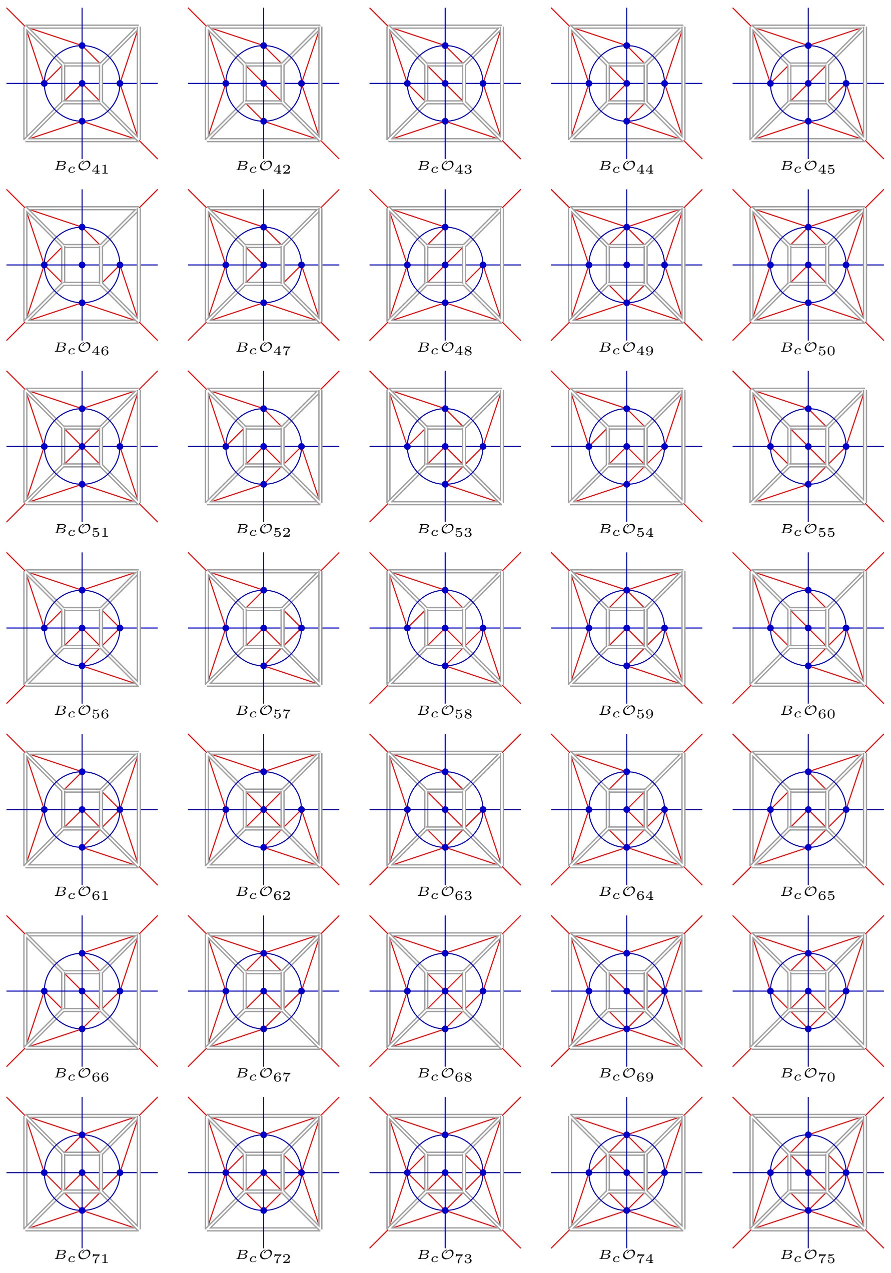


Figure 5 The c-edge assignments in $B\mathcal{O}$ up to isomorphism (cont.)

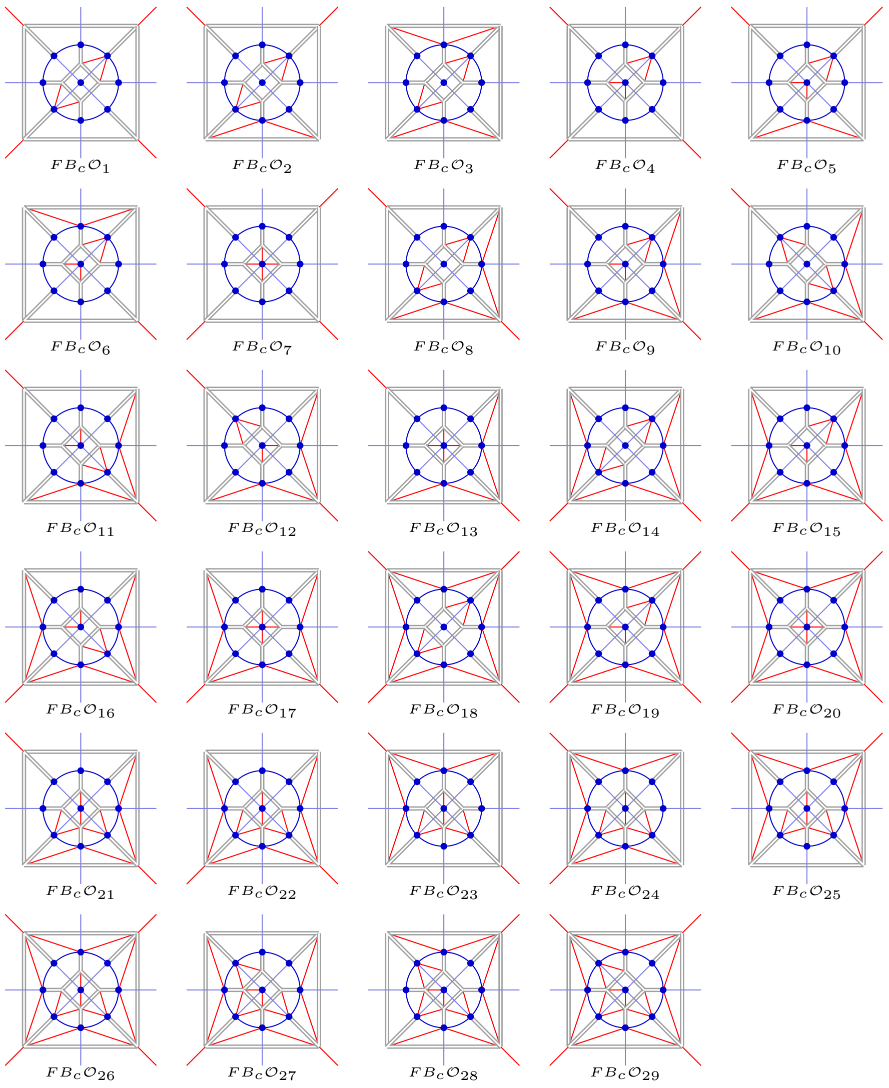


Figure 6 The c-edge assignments in $FB\mathcal{O}$ up to isomorphism

References

1 C. P. Avelino, A. F. Santos, Right triangular dihedral f–tilings of the sphere: $\left(\alpha,\beta,\frac{\pi}{2}\right)$ and $\left(\gamma,\gamma,\frac{\pi}{2}\right)$, *Ars Combinatoria* **121** (2015), 227–274.

2 C. P. Avelino, A. F. Santos, Triangular spherical dihedral f-tilings: the ($\pi/2$, $\pi/3$, $\pi/4$) and ($2\pi/3$, $\pi/4$, $\pi/4$) family, *Rev. Union Mat. Argent.* **61** (2020), 367–387.

3 Y. Akama, E. X. Wang, M. Yan, Tilings of the sphere by congruent pentagons III: edge combination a^5, *Adv. in Math.* **394** (2022), #107881.

4 Y. Akama, M. Yan, On deformed dodecahedron tiling, *Australas. J. of Comb.* **85(1)** (2023), 1–14.

5 L. Bieberbach, Über die Bewegungsgruppen des n-dimensionalen euklidisches Raumes mit einem endlichen Fundamentalbereich, *Gött. Nachr.* (1910), 75–84.

6 H. M. Cheung, H. P. Luk, M. Yan, Tilings of the sphere by congruent quadrilaterals or triangles, *preprint*, 2022, `arXiv:2204.02736`.

7 H. M. Cheung, H. P. Luk, M. Yan, Tilings of the sphere by congruent pentagons IV: edge combination a^4b, *preprint*, 2023, `arXiv:2307.11453`.

8 E. S. Fedorov, Symmetry in the plane, *Proc. Imperial St. Petersburg Mineral. Soc.* **28** (1891), 345–390 (in Russian).

9 J. B. Fraleigh, Section 16, in: *A First Course in Abstract Algebra*, Pearson, 2014, p.158.

10 B. Grünbaum, G. C. Shephard, The 81 types of isohedral tilings of the plane, *Math. Proc. Cambridge Philos. Soc.* **82** (1977), 177–196.

11 H. H. Gao, N. Shi, M. Yan, Spherical tiling by 12 congruent pentagons, *J. Comb. Theory Ser. A* **120(4)** (2013), 744–776.

12 H. Heesch, Aufbau der Ebene aus kongruenten Bereiche, *Nachr. Ges. Wiss. Gött.* (1935), 115–117.

13 H. P. Luk, H. M. Cheung, Rational angles and tilings of the sphere by congruent quadrilaterals, *Ann. Comb.* **28** (2024), 485–527.

14 G. D. Mostow, Braids, hypergeometric functions, and lattices, *Bulletin of the American Mathematical Society* **16(2)** (1987), 225–246.

15 R. Nedela, M. Škoviera, Maps, in: *Handbook of Graph Theory*, 2nd. Ed., CRC Press, 2014, 826–827.

16 G. Pólya, Über die Analogie der Kristallsymmetrie in der Ebene, *Z. Krist.* **60** (1924), 278–282.

17 K. Reinhardt, Zur Zerlegung Euklische Räume in kongruente Polytope, *Sitzungsber. Preuss. Akad. Wiss.* (1928), 150–155.

18 R. Penrose, Role of aesthetics in pure and applied research, *Bulletin of the Institute of Mathematics and Its Applications* **10(266)** (1974), 266–271.

19 S. A. Robertson, Isometric folding of Riemannian manifolds, *Proc. R. Soc. Edinb.* **79** (1977), 275–284.

20 M. J. Sewell, Complementary energy and catastrophes, *Technical Summary Report*, University of Wisconsin, Madison, Math Research Center, 1978.

21 J. Širáň, Triangle group representations and their applications to graphs and maps, *Discrete Mathematics* **229**, (2001) 341–358.

22 D. Smith, J. S. Myers, C. S. Kaplan, C. Goodman-Strauss, An aperiodic monotile, *Combinatorial Theory* **4(1)** (2024), #6.

23 Y. Ueno, Y. Agaoka, Classification of tilings of the 2-dimensional sphere by congruent triangles, *Hiroshima Math. J.* **32(3)** (2002), 463–540.

24 E. X. Wang, M. Yan, Tilings of sphere by congruent pentagons I: edge combinations a^2b^2c and a^3bc, *Adv. in Math.* **394** (2022), #107866.

25 E. X. Wang, M. Yan, Tilings of sphere by congruent pentagons II: edge combination a^3b^2, *Adv. in Math.* **394** (2022), #107867.

Balanced TSP partitioning

Benjamin Aram Berendsohn[1], Hwi Kim[2], and László Kozma[3]

1 **Max Planck Institute for Informatics, Saarbrücken, Germany. Work supported by DFG grant KO 6140/1-2.**
`benjamin.berendsohn@fu-berlin.de`

2 **Department of Computer Science and Engineering, Pohang University of Science and Technology, Republic of Korea.**
`hwikim@postech.ac.kr`

3 **Institut für Informatik, Freie Universität Berlin, Germany. Work supported by DFG grant KO 6140/1-2.**
`laszlo.kozma@fu-berlin.de`

Abstract

The *traveling salesman problem* (TSP) famously asks for a shortest tour that a salesperson can take to visit a given set of cities in any order. In this paper, we ask how much faster $k \geq 2$ salespeople can visit the cities if they divide the task among themselves. We show that, in the two-dimensional Euclidean setting, two salespeople can always achieve a speedup of at least $\frac{1}{2} + \frac{1}{\pi} \approx 0.818$, for any given input, and there are inputs where they cannot do better. We also give (non-matching) upper and lower bounds for $k \geq 3$.

1 Introduction

The *traveling salesman problem* (TSP) asks, given n cities and their pairwise distances, for the shortest tour that visits each city. It is one of the best studied optimization problems, well-known to be NP-hard even in the planar Euclidean case, i.e., if the cities are points in $\mathbb{R}^2$, and the distance between any two points is the Euclidean distance [11]. This restricted version of the problem admits a polynomial-time approximation scheme (PTAS) [3, 14], whereas the more general case of metric distances is known to admit an approximation ratio slightly below 1.5 (from the recent breakthrough [13] that improved the longstanding approximation ratio of 1.5 due to Christofides [8]).

In this paper, we focus on the two-dimensional Euclidean case and study the following natural question: How much faster can all cities be visited if multiple salespeople can collaborate on performing the task, and each city has to be visited by at least one salesperson? More formally, we wish to cover a given point set P with multiple (say, k) closed curves, instead of a single one, minimizing the maximum length among the k curves. Intuitively, the k salespeople execute their tours simultaneously, and all have to return to their respective (arbitrary) starting points before a given deadline.

The specific question we ask is: How does the cost improve with the parameter k, when compared to the normal TSP cost, in the worst case? This ratio, which we precisely define next, can be seen as an inherent measure of *decomposability* of the TSP problem.

A *tour* of a point set $P \subset \mathbb{R}^2$ is a closed polygonal curve that contains each point in P. A tour is *optimal* if its length is minimal among all tours.[1] The length of an optimal tour of P is denoted by $\mathsf{TSP}(P)$. Note that the optimal tour is necessarily a simple polygon, and in particular, it visits each point in P exactly once. (We still allow degenerate cases, such

[1] We do not require vertices of a tour to be from P, though this is clearly the case for *optimal* tours.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

as polygons with one or two corners, or polygons on multiple collinear points.) Let $\mathsf{S}_k(P)$ denote the set of partitions of P into at most k subsets, and let

$$\mathsf{TSP}_k(P) = \min_{R \in \mathsf{S}_k(P)} \max_{Q \in R} \mathsf{TSP}(Q).$$

Intuitively, the quantity $\mathsf{TSP}_k(P)$ corresponds to the least amount of time k salespeople need to serve the points in P. In particular, $\mathsf{TSP}_1(P) = \mathsf{TSP}(P)$.

The ratio $\gamma(P,k) = \mathsf{TSP}_k(P)/\mathsf{TSP}(P)$ indicates the advantage that k salespeople have over a single one. Observe that always $\gamma(P,k) \leq 1$, but $\gamma(P,k)$ can be arbitrarily small (e.g., if P consists of two very small clusters that are far apart from each other). In consequence, it makes sense to ask how *large* $\gamma(P,k)$ can get, i.e., how much of an improvement we can *guarantee* when using multiple salespeople. Accordingly, we define:

$$\gamma(k) = \sup_P \gamma(P,k) = \sup_P \left(\frac{\mathsf{TSP}_k(P)}{\mathsf{TSP}(P)} \right).$$

Clearly, $\gamma(1) = 1$. The main result of this paper is the precise determination of $\gamma(2)$.

▶ **Theorem 1.1.** $\gamma(2) = \frac{1}{2} + \frac{1}{\pi} \approx 0.818$.

We further give some lower and upper bounds for $\gamma(k)$ when $k \geq 3$.

▶ **Theorem 1.2.** *For all $k \geq 2$, we have $\gamma(k) \geq \frac{1}{k} + \frac{1}{\pi} \sin \frac{\pi}{k}$.*

▶ **Theorem 1.3.** *For all $a, b \in \mathbb{N}$, we have $\gamma(a \cdot b) \leq \gamma(a) \cdot \gamma(b)$.*

▶ **Theorem 1.4.** *For all $a, b \in \mathbb{N}$, we have*

$$\gamma(a+b) \leq \left(1 + \tfrac{2}{\pi}\right) \cdot \frac{\gamma(a) \cdot \gamma(b)}{\gamma(a) + \gamma(b)}.$$

From our results we can derive, e.g., $\gamma(3) \in (0.609, 0.737)$, $\gamma(4) \in (0.475, 0.670)$. For a table of approximate results for larger k, and proof details omitted in this abstract, we refer to the arXiv version of the paper. We leave determining tight bounds for $k \geq 3$ as a challenging open question. A generalization of the problem to higher dimensions or to more general metric spaces is likewise interesting.

Related work. Although many variants of TSP with multiple salespeople have been studied in the literature, we are not aware of the ratio $\gamma(k)$ being explicitly considered before. We mention a few works that study problems of a similar flavor.

The *min-max cycle cover* problem refers to a quantity essentially equivalent to TSP_k in a more general weighted graph setting; e.g., see [17]. In the TSP literature, closely related problems include the *k-person TSP*, *multiple TSP*, and *multi-depot vehicle routing*, e.g., see [4, 16, 15, 6] (and references therein). These problems are studied under various optimization criteria and often with additional constraints; for instance, in several formulations a starting point for each salesperson, or alternatively, a common starting point for all, are specified. A "depot-free" variant essentially matching our setting has been studied in [9]. All these works aim at computing optimal or approximate solutions for given input instances, whereas our concern is the worst-case ratio $\gamma(k)$ in a geometric/Euclidean case.

In a geometric setting, a cost-ratio similar to ours has been studied for the *minimum spanning tree* (MST) problem, but with *sum*, rather than *maximum* criterion, e.g., see [1, 10]. Bereg et al. [5] study a partitioned MST and TSP problem, where starting points are given and the parts must have *equal cardinality.* Arkin et al. [2] and Johnson [12] study optimization problems under a two-partitioning similar to ours, but with a pairing of points given as part of the input, requiring each pair to be separated.

2 Preliminaries

A *curve* C is the image of a continuous function $c\colon [0,\ell] \to \mathbb{R}^2$, for some $\ell \in \mathbb{R}_+$. We call c a *parametrization* of C. If $c(0) = c(\ell)$, then the curve is *closed*. It is sometimes useful to extend the domain of c to $\mathbb{R}_{\geq 0}$; in this case, we set $c(x) = c(x) \bmod \ell$ if $x > \ell$. Observe that this function is still continuous.

We define *polygons* and *polygonal curves* in the usual way. Observe that a polygonal curve C can be parametrized with a function $c\colon [0,\ell] \to \mathbb{R}^2$ such that for two points $c(x)$ and $c(y)$ on the curve, the distance in clockwise direction along the curve between $c(x)$ and $c(y)$ is precisely $|x - y|$. We then call c a *canonical parametrization* of C. We write $|C| = \ell$ for the length of C.

Given a closed curve C and two distinct points p, q on C, we call the line segment pq a *diagonal*, and we denote by $C(p,q)$ the subpath of C from p to q in the positive direction according to the parametrization. Formally, let $c\colon [0,\ell] \to \mathbb{R}^2$ be a parametrization of C and let $p = c(x_1)$ and $q = c(x_2)$. Then $C(p,q)$ is the curve parametrized by c', where c' is c restricted to the interval $[x_1, x_2]$ if $x_1 < x_2$, or to the interval $[x_2, \ell + x_1]$ otherwise.

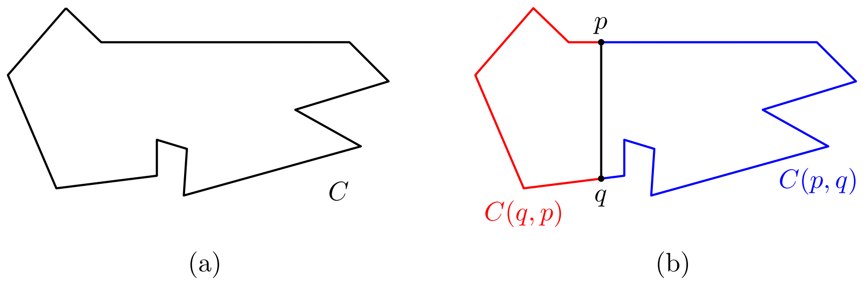


Figure 1 (a) A curve C. (b) Curves $C(p,q)$ and $C(q,p)$, assuming a clockwise parametrization.

Width. Given an angle θ, let $u_\theta = (\cos\theta, \sin\theta)$ be the unit vector in direction θ. The *(directional) width* of a closed curve P in direction θ is the minimum distance between two parallel lines, orthogonal to u_θ, that enclose P (see fig. 2). Equivalently, it is the length of the projection of P to a line parallel to u_θ. Formally, we write:

$$w(u_\theta, P) = \max_{p \in P} \langle u_\theta, p\rangle - \min_{p \in P} \langle u_\theta, p\rangle.$$

The *width* of a closed curve P is the minimum width over all directions:

$$w(P) = \min_{\theta \in [0,\pi)} w(u_\theta, P).$$

It is well known [7] that the *mean width* (computed by integration over all directions) of a closed convex curve is precisely its length divided by π. Since the width of a curve is clearly at most its mean width, we have:

▶ **Lemma 2.1.** *For any convex closed curve C on $\mathbb{R}^2$, $w(C) \leq |C|/\pi$.*

3 Lower bounds: the circular point set

We start with an illustrative special case that will also provide our lower bound for the case $k = 2$. Let P_n be a set of $n \geq 2$ points, arranged regularly-spaced along the unit circle. More precisely, define $P_n = \{p_i \mid 1 \leq i \leq n\}$ with $p_i = (\cos(2\pi\frac{i}{n}), \sin(2\pi\frac{i}{n}))$. See fig. 3 (a).

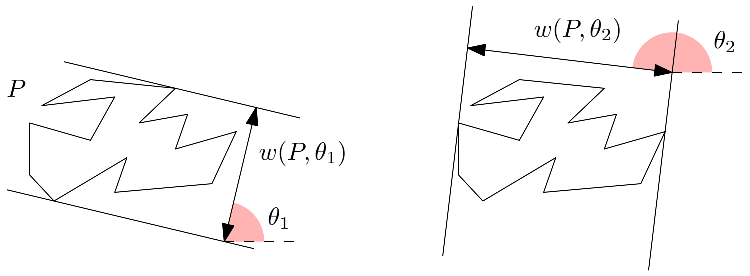


Figure 2 A polygon and its width with respect to directions θ_1 and θ_2. Here $w(P, \theta_1) < w(P, \theta_2)$.

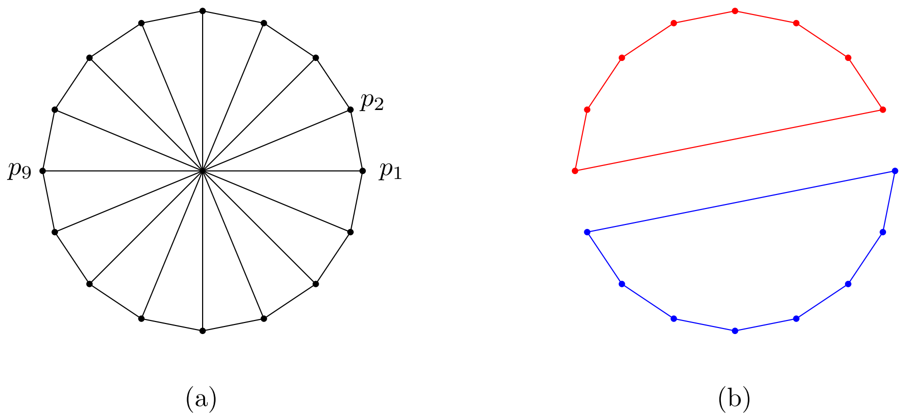


Figure 3 (a) Point set P_{16}. (b) Balanced partition of P_{16}.

Clearly, the shortest tour of P_n goes around the circle, approaching a length of 2π as n goes to infinity. Thus, we have $\mathsf{TSP}(P_n) \to 2\pi$. Now, consider the case $k = 2$. The intuitively best way of splitting the point set is shown in fig. 3 (b): Divide the circle with a straight cut into equal parts (assume for simplicity that n is even). For both parts, take the half-circle plus the diagonal as a tour. As n goes to infinity, the length of each of these tours will tend towards $\pi + 2$ (half the circumference plus twice the radius). Thus, the following holds:

$$\lim_{n\to\infty} \gamma(P_n, 2) \leq \tfrac{\pi+2}{2\pi} = \tfrac{1}{2} + \tfrac{1}{\pi}.$$

Later, in section 4, we show that this upper bound holds for *every* point set, using a similar technique of “halving” the optimal tour.

For our circular point set, it turns out that the bound is tight, since the above construction is optimal, as we show now. The main technical lemma is the following (proof omitted).

▶ **Lemma 3.1.** *Let* $P_{2n} = \{p_i \mid 1 \leq i \leq 2n\}$ *with* $p_i = (\cos(2\pi\frac{i}{2n}), \sin(2\pi\frac{i}{2n}))$. *Let* $m \leq 2n$*, let* $A_m = \{p_1, p_2, \dots, p_m\}$*, and let* $B \subseteq P_{2n}$ *be an arbitary subset of size* m*. Then* $\mathsf{TSP}(A_m) \leq \mathsf{TSP}(B)$.

Let us argue how it implies our claim. Take any partition of P_{2n} into two sets A and B. Without loss of generality, we have $|B| \geq n$. Thus, by Lemma 3.1, we have $\mathsf{TSP}(B) \geq \mathsf{TSP}(A_n)$. Therefore, the partition $\{A_n, P_{2n} \setminus A_n\}$ is optimal, as desired. As argued above, we have $\lim_{n\to\infty} \mathsf{TSP}(A_n)/\mathsf{TSP}(P_{2n}) = \frac{1}{2} + \frac{1}{\pi}$. Hence, we have $\gamma(P_{2n}, 2) \to \frac{1}{2} + \frac{1}{\pi}$, and thus $\gamma(2) \geq \frac{1}{2} + \frac{1}{\pi}$. This proves the lower bound of Theorem 1.1.

Using Lemma 3.1, it is not hard to extend the lower bound to larger k, by considering the set P_{kn} and its partition into k sets of n consecutive points on the circle.

▶ **Theorem 1.2.** *For all $k \geq 2$, we have $\gamma(k) \geq \frac{1}{k} + \frac{1}{\pi}\sin\frac{\pi}{k}$.*

4 Upper bounds via short diagonals

We now prove several upper bounds for $\gamma(k)$. Recall that to show $\gamma(k) \leq \alpha$, we need to argue that for *every* point set P, there is a partition into k subsets $Q_1, \ldots, Q_k$ with $\mathsf{TSP}(Q_i) \leq \alpha \cdot \mathsf{TSP}(P)$ for all $i \in [k]$.

Our upper bound for $\gamma(2)$ adapts the idea for the circular point set from section 3. Essentially, we take the optimal tour C of P, split it in two at some point p and its antipodal point q, and split the point set accordingly into Q_1, Q_2. Let $C_1 = C(p,q) \cup pq$ and $C_2 = C(q,p) \cup pq$. Observe that C_1, resp. C_2 are tours of Q_1, resp. Q_2, and $|C_1| = |C_2| = |C|/2 + |pq|$. It turns out that we can always find an antipodal pair p, q on C such that the diagonal $|pq|$ is short.

▶ **Lemma 4.1.** *Let C be a closed polygonal curve. Then, there exists a diagonal pq of C such that $|C(p,q)| = |C|/2$ and $|pq| \leq \frac{1}{\pi}|C|$.*

Proof sketch. Let H be the convex hull of C. Then the width of H is at most $\frac{1}{\pi}|H| \leq \frac{1}{\pi}|C|$, by Lemma 2.1. This means that there exists a direction θ such that the width of H in direction θ is at most $\frac{1}{\pi}|C|$. Clearly, the same is true for C, so any diagonal of C that is parallel to u_θ has length at most $\frac{1}{\pi}|C|$. Finally, for *every* direction θ, there exists an antipodal pair p, q such that the diagonal pq is parallel to u_θ. (Essentially, we can rotate p around the curve, obtaining every possible direction.) For a full proof, see Lemma 4.3 below. ◀

Using Lemma 4.1, the above discussion yields a partition of P into two sets Q_1, Q_2 with $\mathsf{TSP}(Q_1), \mathsf{TSP}(Q_2) \leq \frac{1}{2}|C| + \frac{1}{\pi}|C|$. Since $\mathsf{TSP}(P) = |C|$ by assumption, we have $\gamma(2) \leq \frac{1}{2} + \frac{1}{\pi}$. This concludes the proof of Theorem 1.1.

Our second upper bound is a simple reduction for non-prime k.

▶ **Theorem 1.3.** *For all $a, b \in \mathbb{N}$, we have $\gamma(a \cdot b) \leq \gamma(a) \cdot \gamma(b)$.*

Proof. Let P be a point set. By definition, there is a partition of P into sets $Q_1, Q_2, \ldots, Q_a$ such that $\mathsf{TSP}(Q_i) \leq \gamma(a) \cdot \mathsf{TSP}(P)$ for each $i \in [a]$. We can now further split each Q_i into b sets $Q_{i,1}, Q_{i,2}, \ldots, Q_{i,b}$, for a total of $a \cdot b$ sets with $\mathsf{TSP}(Q_{i,j}) \leq \gamma(b) \cdot \mathsf{TSP}(Q_i) \leq \gamma(b) \cdot \gamma(a) \cdot \mathsf{TSP}(P)$. ◀

Our third upper bound combines the short-diagonal technique with the recursive approach of Theorem 1.3. The key is a generalization of Lemma 4.1. Essentially, we can always find a short diagonal to split the curve into not only two halves, but into two parts of *any chosen length.* As a first step, we show that for every prescribed curve length and direction, we can find an appropriate diagonal.

▶ **Lemma 4.2.** *Let $c\colon [0,1] \to \mathbb{R}^2$ be the parametrization of a closed curve, let $x \in \mathbb{R}$, $0 < x < 1$, and let u be a vector. Then there exists some $t \in [0,1]$ such that the vector $c(t+x) - c(t)$ is orthogonal to u.*

Proof. Define $v(t) = c(t+x) - c(t)$ and $f(t) = v(t) \cdot u$. We need to show that there is some $t \in [0,1]$ such that $f(t) = 0$.

Let $p_0 = c(t_0)$ be a point on c minimizing $p_0 \cdot u$. See fig. 4. Observe that p_0 exists, since the image of c is compact. We claim that $f(t_0) \geq 0$. Indeed, if $f(t_0) < 0$, then, by definition:

$$(c(t_0+x) - c(t_0)) \cdot u < 0 \iff c(t_0+x) \cdot u < c(t_0) \cdot u,$$

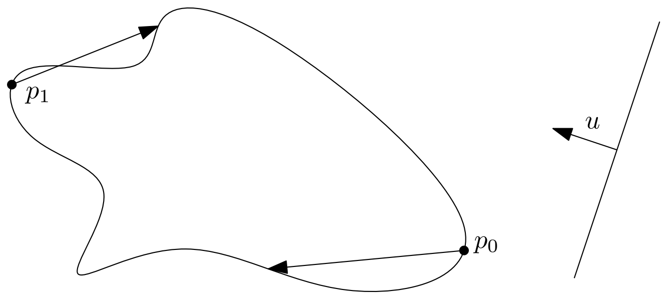


Figure 4 Illustration of Lemma 4.2.

which contradicts minimality of $c(t_0) \cdot u$.

Similarly, let $p_1 = c(t_1)$ minimize $p_1 \cdot u$. By symmetry, we have $f(t_1) \leq 0$. Finally, since f is continuous, there must be some t with $f(t) = 0$ by the intermediate value theorem. This concludes the proof. ◀

We now show how to split curves into parts of a prescribed length.

▶ **Lemma 4.3.** *Let C be a closed polygonal curve. Then, for each given $x \in \mathbb{R}$, $0 < x < |C|$, there exists a diagonal pq of C such that $|C(p, q)| = x$ and the length of pq is at most $\frac{1}{\pi}|C|$.*

Proof. Let c be a canonical parametrization of c and let H be the convex hull of C. Let θ be the angle that minimizes the directional width of H, i.e., we have $w(\theta, H) = w(H)$. By Lemma 2.1, we have $w(H) \leq \frac{1}{\pi}|H| \leq \frac{1}{\pi}|C|$.

Now let u be a vector orthogonal to u_θ. By Lemma 4.2, there exists a $t \in [0, |C|]$ such that $c(t + x) - c(t)$ is orthogonal to u, and thus parallel to u_θ. Let $p = c(t)$ and $q = c(t + x)$. Then $|C(p, q)| = x$ by definition, and the length of pq is bounded by $w_\theta(C) \leq w_\theta(H) \leq \frac{1}{\pi}|C|$. This concludes the proof. ◀

Using Lemma 4.3, we obtain the following generic upper bound:

▶ **Lemma 4.4.** *Let $k, a, b \in \mathbb{N}$ with $k = a + b$, and let $x \in \mathbb{R}$, $0 < x < 1$. Then we have $\gamma(k) \leq \max\left((x + \frac{1}{\pi})\gamma(a), (1 - x + \frac{1}{\pi})\gamma(b)\right)$.*

Proof. Let P be a point, and T be a tour of P; without loss of generality, we have $|T| = 1$.

We first use Lemma 4.3 to obtain a diagonal pq of length at most $\frac{1}{\pi}$ such that $|C(p, q)| = x$. As in the proof of the upper bound of Theorem 1.1, we take the two tours $C_1 = C(p, q) \cup pq$ and $C_2 = C(q, p) \cup pq$, with $|C_1| \leq x + \frac{1}{\pi}$ and $|C_2| \leq 1 - x + \frac{1}{\pi}$.

Now partition P into P_1, P_2 such that $P_1 \subseteq C_1$ and $P_2 \subseteq C_2$. By definition, we have $\mathsf{TSP}(P_1) \leq x + \frac{1}{\pi}$ and $\mathsf{TSP}(P_2) \leq 1 - x + \frac{1}{\pi}$. Optimally partitioning P_1 into a parts will yield tours of length at most $\gamma(a)\,\mathsf{TSP}(P_1)$, by definition, and similarly, optimally partitioning P_2 into b parts will yield tours of length $\gamma(b)\,\mathsf{TSP}(P_2)$. Overall, we have $a + b = k$ tours of length at most $\max(\gamma(a)\,\mathsf{TSP}(P_1), \gamma(b)\,\mathsf{TSP}(P_2))$, as desired. ◀

Optimizing for x in Lemma 4.4 yields

$$x = \frac{\gamma(b)}{\gamma(a) + \gamma(b)} + \frac{\gamma(b) - \gamma(a)}{\pi \cdot (\gamma(a) + \gamma(b))}.$$

Finally, with a simple calculation, we have:

▶ **Theorem 1.4.** *For all $a, b \in \mathbb{N}$, we have*

$$\gamma(a+b) \leq \left(1+\tfrac{2}{\pi}\right) \cdot \frac{\gamma(a) \cdot \gamma(b)}{\gamma(a)+\gamma(b)}.$$

Note that the upper bound of Theorem 1.1 follows from the case $a = b = 1$ and $\gamma(1) = 1$.

References

1 Afrouz Jabal Ameli, Faezeh Motiei, and Morteza Saghafian. The complexity of maximizing the MST-ratio. *arXiv preprint*, 2024. arXiv:2409.11079.

2 Esther M. Arkin, Aritra Banik, Paz Carmi, Gui Citovsky, Su Jia, Matthew J. Katz, Tyler Mayer, and Joseph S. B. Mitchell. Network Optimization on Partitioned Pairs of Points. In *Proceedings of the 28th International Symposium on Algorithms and Computation (ISAAC)*, pages 6:1–6:12, 2017. doi:10.4230/LIPIcs.ISAAC.2017.6.

3 Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998. doi:10.1145/290179.290180.

4 Tolga Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *omega*, 34(3):209–219, 2006. doi:10.1016/J.OMEGA.2004.10.004.

5 Sergey Bereg, Yuya Higashikawa, Naoki Katoh, László Kozma, Manuel Lafond, Günter Rote, Yuki Tokuni, Max Willert, and Binhai Zhu. The two-squirrel problem and its relatives. *To appear*, 2025.

6 Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & industrial engineering*, 99:300–313, 2016.

7 A. Cauchy. Note sur divers théorèms relatifs à la rectification des courbes et à la quadrature des surfaces. *Comptes rendus de l'Académie des Sciences*, 13:1060–1065, 1941.

8 Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Technical Report 388, Carnegie Mellon University, 1976.

9 José Alejandro Cornejo-Acosta, Jesús García-Díaz, Julio César Pérez-Sansalvador, and Carlos Segura. Compact integer programs for depot-free multiple traveling salesperson problems. *Mathematics*, 11(13):3014, 2023.

10 Adrian Dumitrescu, János Pach, and Géza Tóth. Two trees are better than one. *arXiv preprint*, 2023. arXiv:2312.09916.

11 Michael R. Garey and David S. Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

12 Matthew Johnson. Red-Blue-Partitioned MST, TSP, and Matching. In *Proceedings of the 30th Canadian Conference on Computational Geometry (CCCG)*, 2018.

13 Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric tsp. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 32–45, 2021. doi:10.1145/3406325.3451009.

14 Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999. doi:10.1137/S0097539796309764.

15 Jairo R. Montoya-Torres, Julián López Franco, Santiago Nieto Isaza, Heriberto Felizzola Jiménez, and Nilson Herazo-Padilla. A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering*, 79:115–129, 2015.

16 Zhou Xu and Brian Rodrigues. A 3/2-approximation algorithm for multiple depot multiple traveling salesman problem. In *Algorithm Theory-SWAT 2010: 12th Scandinavian Symposium and Workshops on Algorithm Theory, Bergen, Norway, June 21-23, 2010. Proceedings 12*, pages 127–138. Springer, 2010. doi:10.1007/978-3-642-13731-0_13.

17 Wei Yu and Zhaohui Liu. Improved approximation algorithms for some min-max and minimum cycle cover problems. *Theoretical Computer Science*, 654:45–58, 2016. doi:10.1016/j.tcs.2016.01.041.

On the Non-Locality of Edge Insertions

Florestan Brunck[1]

1 **University of Copenhagen, Denmark**
flbr@di.ku.dk

Abstract

We challenge the idea that edge insertions are local improvement operations and show that the edge-insertion algorithm must sometimes insert an edge between vertices that are at the farthest combinatorial distance apart, and that this edge must cross linearly many edges of the triangulation for the algorithm to escape a local optimum and return the optimal triangulation.

1 Introduction

The edge insertion algorithm introduced by Edelsbrunner et al. [1, 2] provides an escape from the local optima where regular edge flipping algorithms for triangulations tend to get stuck. It generalises the usual elementary diagonal flip in quadrilaterals to the more powerful operation of inserting an edge between any two vertices in the triangulation, followed by the retriangulation of the polygonal region left after removing all edges of the original triangulation which crossed the inserted edge. It currently offers the best polynomial algorithms for a number of optimisation problems, such as the min-max angle problem ([2]). While flips are inherently local elementary operations, edge insertions are not. It is then natural to wonder exactly how non-local edge insertions have to be to guarantee optimisation. Perhaps surprisingly - especially in contrast with the closely related max-min problem where the Delaunay triangulation can be reached via flips - we show that edge insertions cannot be constrained to be local operations at all when trying to optimise triangulations for the min-max angle problem. In particular, we give an example where the edge insertion algorithm must insert an edge between vertices that are at the farthest combinatorial distance apart, and that this edge must also cross linearly many edges for the edge insertion algorithm to escape a local optimum and return the min-max angle triangulation.

▶ **Theorem 1.1.** *For any integer $n \in \mathbb{N}$, there exists a triangulation $\mathcal{T}_n$ with $4n+5$ edges and combinatorial diameter $n+2$, such that, in order to successfully return the min-max angle triangulation, the edge insertion algorithm must insert an edge between vertices which are at combinatorial distance $n+2$. Furthermore, that edge intersects $2n$ edges of $\mathcal{T}_n$.*

2 Background - The Edge Insertion Algorithm

Recall that a *triangulation* of a finite set of points $\mathcal{S}$ in the Euclidean plane is a maximally connected plane graph with set of vertices $\mathcal{S}$. We denote by μ the function that maps a triangle t in $\mathcal{T}$ to the value of its maximum angle. The example we provide concerns the *min-max angle problem*, namely the task of finding the triangulation of a fixed point set in the plane which minimises the maximum value of μ over all triangles t in the triangulation. The *measure* $\mu(\mathcal{T})$ of a triangulation $\mathcal{T}$ is defined as the quantity $\max\{\mu(t) \mid t \text{ a triangle in } \mathcal{T}\}$. If $\mathcal{T}$ and $\mathcal{T}'$ are two triangulations of the same point set, we say that $\mathcal{T}$ is an improvement of $\mathcal{T}'$ and write $\mathcal{T} \prec \mathcal{T}'$ if $\mu(\mathcal{T}) < \mu(\mathcal{T}')$ or if $\mu(\mathcal{T}) = \mu(\mathcal{T}')$ and the set of triangles t in $\mathcal{T}$ such that $\mu(t) = \mu(\mathcal{T})$ is a strict subset of the set of such triangles in $\mathcal{T}'$. A triangulation $\mathcal{T}$ is *optimal* if there is no improvement of $\mathcal{T}$.

Given a triangulation $\mathcal{T}$ of a point set $\mathcal{S}$ in the plane, the *edge insertion* UV, for $U, V \in \mathcal{S}$ is the following procedure (Fig. 1):

Algorithm 1

Edge Insertion($\mathcal{T}, UV$)

```
    $\mathcal{T}' \longleftarrow \mathcal{T}$
    Add the edge $UV$ and remove from $\mathcal{T}'$ all edges that intersect $UV$
    *This creates two polygonal regions $L$ and $R$ on either side of the inserted edge*
    Retriangulate $L$ and $R$ optimally
    *This can be done by brute force or by dynamic programming when possible*
    Return $\mathcal{T}'$
```

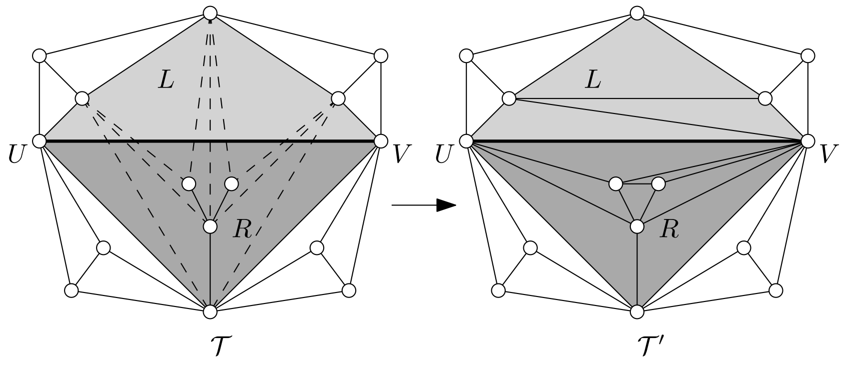


Figure 1 : The edge insertion of an edge UV (in bold) in a triangulation $\mathcal{T}$ of a planar point set. The original edges of $\mathcal{T}$ which intersect UV are shown in dashed lines on the leftmost diagram while the optimally retriangulated ones are shown on the rightmost diagram. The affected regions L and R are highlighted in light and dark grey.

The *edge insertion algorithm* is then the following:

Algorithm 2

Edge Insertion Algorithm($\mathcal{S}$)

```
    *Input: $\mathcal{S}$ a planar point set*
    *Output: an optimal triangulation $\mathcal{T}_{opt}$ for $\mathcal{S}$*
    Construct an arbitrary triangulation $\mathcal{T}$ of $\mathcal{S}$
    **repeat** $\mathcal{T}' \longleftarrow \mathcal{T}$
        **for** all edges $UV$ with $U \neq V \in \mathcal{S}$ **do**
            $\mathcal{T}'' :=$ Edge Insertion($\mathcal{T}, UV$)
            **if** $\mathcal{T}'' \prec \mathcal{T}$ **then** $\mathcal{T} \longleftarrow \mathcal{T}''$ and **exit the for-loop**
            **end if**
        **end for**
    **until** $\mathcal{T} = \mathcal{T}'$
```

The correctness of the above algorithm was proven in [2]. By being more parsimonious in which candidate edge insertions should be considered, and getting rid of dynamic programming for the retriangulation step in favour of a smarter iterated ear-removal process, the following running time was obtained:

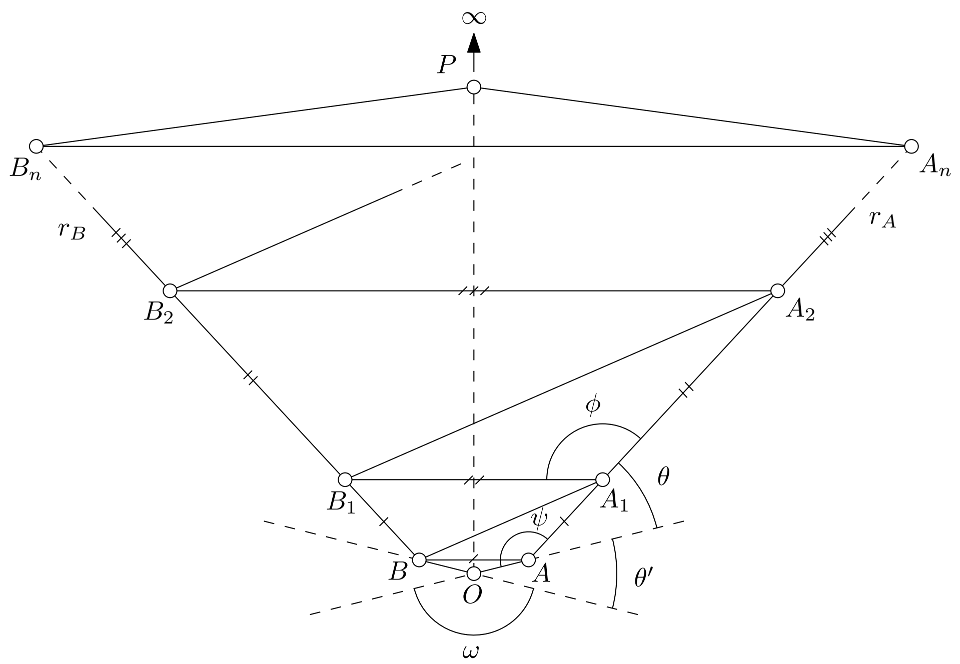
▶ **Theorem 2.1** (Theorem (Edelsbrunner, Waupotitsch and Tan)). *Given a finite set of points $\mathcal{S}$ in the plane, the edge insertion algorithm returns the optimal min-max angle triangulation in time $O(|\mathcal{S}|^2 \log |\mathcal{S}|)$.*

3 The Manta Ray Triangulation

In this section, we describe the construction of our pathological triangulation and prove Theorem 1.1.

Proof of Theorem 1.1. We describe the construction for $\mathcal{T}_n$, as seen on Fig. 2. Starting with an isosceles triangle OAB, consider the ray r_A (resp. r_B) starting at A (resp. B) and meeting the line OA (resp. OB) with a clockwise (resp. counterclockwise) angle of θ. The direction of the rays is chosen such that, for any point R lying on r_A (resp. r_B), the triangle ARB is direct. Let us fix $n \in \mathbb{N}$. We inductively construct points $A_0, A_1, \ldots, A_n$ and $B_0, B_1, \ldots, B_n$ in the following way:

- $A_0 = A$, $B_0 = B$.
- For all $0 \leq i \leq n$, $|A_iA_{i+1}| = |B_iB_{i+1}| = |A_iB_i|$.



21 **Figure 2** : The “manta ray” triangulation $\mathcal{T}_n$.

We additionally define a point P, lying on the perpendicular bisector of A and B on the same side of the line AB as A_1, “far away” from A (exactly how far will be made precise shortly). We can then consider the triangulation containing the triangles OAB, A_nPB_n, as well as the edges (A_i, B_i), (A_{i+1}, B_i), (A_i, A_{i+1}) and (B_i, B_{i+1}), for $0 \leq i \leq n-1$. We readily check that this triangulation has $4n + 5$ edges and combinatorial diameter $n + 2$. Let us introduce some additional notation for the angles. By construction, for $1 \leq i \leq n-1$, the angles $\angle A_{i+1}A_iB_i$ are all equal, let us call this angle ϕ. The triangle angle at O in OAB will be denoted by ω and the angle $\angle A_1AO$ by ψ. Call θ the complementary angle $\pi - \psi$.

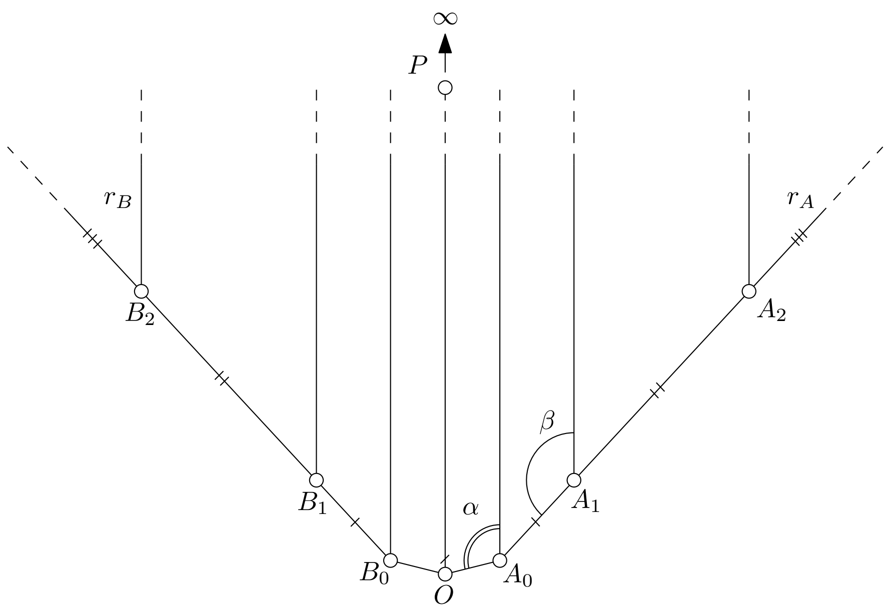
We also denote by α the largest angle between the line OA and the perpendicular to AB passing through A (see Fig. 3). By construction, for all $1 \leq i \leq n$, the angle between the ray r_A and the perpendicular to the line AB passing through A_i does not depend on i. We denote that angle by β. Note that this triangulation has three degrees of freedom and can be fully parametrised by P, ω and θ, so that we may refer to it as $\mathcal{T}_n(P, \omega, \theta)$.

The proof of the theorem follows from the following claim after taking $\mathcal{T}_n \coloneqq \mathcal{T}_n(P, \omega_0, \theta_0)$:

▶ Claim 1. There exists a position of P and a value of ω_0 and θ_0 such that the following inequalities hold in the triangulation $\mathcal{T}_n(P, \omega_0, \theta_0)$:

(i) $\phi < \omega$
(ii) $\psi > \omega$
(iii) $\alpha < \omega$
(iv) $\beta < \omega$
(v) $\angle A_n P B_n < \omega$

Inequality (i) and (v) show that ω is the largest angle in $\mathcal{T}_n$. Thus inserting any edge which does not contain O cannot yield an improvement. Inequality (ii) shows that any edge insertion that does not use the edge OP does not yield an improvement of $\mathcal{T}_n$. Indeed, any edge OX with $X = A_i$ or $X = B_i$, $1 \leq i \leq n$, forces the existence of the triangle OAA_1 or OBB_1 in the triangulation. Finally, inequalities (iii) and (iv) show that inserting the edge OP and re-triangulating by joining P to all the other vertices indeed yields an improvement of $\mathcal{T}_n$ (in fact, the min-max angle triangulation itself). Indeed, for any choice of P, we have both $\angle OAP < \alpha$ and $\angle AA_iP < \alpha$ so these last two inequalities hold also for all the angles $\angle OAP$ and $\angle AA_iP$.



22 **Figure 3** : The optimal triangulation minimising the maximum angle.

Proof of the claim. Let θ be the smallest angle between the lines OA and AA_1. We first eliminate a degree of freedom by adding the constraint $\theta = \frac{2}{3}\theta'$ (see Fig. 2). Through straightforward angle-chasing, one can then check that enforcing $\omega > \frac{10}{13}\pi$ together with (v) gives us suitable values for the claim and we need only fix any ω_0 in that range (which then also fixes the value of θ_0). ◀

◀

▶ Remark. If we want the points to be in general position, we can simply perturb the A_i (resp. B_i) to lie on a concave curve by nudging A_1 (resp B_1) slightly towards the inside of the basket (to ensure the triangles OAA_1, OBB_1 are still needed) and nudging A_{i+1} so that the segment $A_{i-1}A_{i+1}$ lies outside of the basket (similarly for each B_i).

Acknowledgements

I am indebted to insightful discussions with Herbert Edelsbrunner, which sparked my curiosity and prompted this observation.

References

1 M. Bern, H. Edelsbrunner, D. Eppstein, S. Mitchell, and T.S. Tan. Edge insertion for optimal triangulations. *Discrete & Computational Geometry*, 10:47–65, 1993.

2 H. Edelsbrunner, T.S. Tan, and R. Waupotitsch. An $O(n^2 \log n)$ time algorithm for the minmax angle triangulation. In *Proceedings of the sixth annual symposium on Computational geometry*, pages 44–52, 1990.

The Induced Matching Distance: A Novel Topological Metric with Applications in Robotics

Javier Perera-Lago[1], **Álvaro Torras-Casas**[2], **Jérôme Guzzi**[3], and **Rocio Gonzalez-Diaz**[1]

1 **Universidad de Sevilla, Seville, Spain,** `{jperera,rogodi}@us.es`
2 **Inserm, INRAE, Centre for Research in Epidemiology and Statistics (CRESS), Université Paris Cité and Sorbonne Paris Nord,** `alvaro.torras-casas@inserm.fr`
3 **SUPSI, IDSIA, Lugano, Switzerland,** `jerome.guzzi@idsia.ch`

Abstract

This paper introduces the induced matching distance, a novel topological metric designed to compare discrete structures represented by a symmetric non-negative function. We apply this notion to analyze agent trajectories over time. We use dynamic time warping to measure trajectory similarity and compute the 0-dimensional persistent homology to identify relevant connected components, which, in our context, correspond to groups of similar trajectories. To track the evolution of these components across time, we compute induced matching distances, which preserve the coherence of their dynamic behavior. We then obtain a 1-dimensional signal that quantifies the consistency of trajectory groups over time. Our experiments demonstrate that our approach effectively differentiates between various agent behaviors, highlighting its potential as a robust tool for topological analysis in robotics and related fields.

1 Persistence Matching Distance Induced by Bijections

From this point forward, we consider sets of n points $Z = \{z_1, z_2, \dots, z_n\}$ together with a symmetric non-negative function $d_Z\colon Z \times Z \to \mathbb{R}^+$, where $\mathbb{R}^+$ is the set of non-negative real numbers. Here, we remark that d_Z might admit zero values between different points and does not need to satisfy the triangle inequality; as this is the case with the dynamic time warping measure considered in Section 2. Since we are only interested in the 0-dimensional persistent homology, we work with the 1-skeleton of the Vietoris-Rips filtration of Z and we fix $\mathbb{Z}_2$ as the ground field.

The *1-skeleton of the Vietoris-Rips filtration* of Z, denoted by $\mathrm{VR}(Z)$, is a family of graphs $\{\mathrm{VR}_r(Z)\}_{r\in\mathbb{R}^+}$ whose vertex set is Z. We set $\mathrm{VR}_0(Z)$ to have no edges, and, for all $r > 0$, each graph $\mathrm{VR}_r(Z)$ contains the edges $[z, z']$ such that $d_Z(z, z') \leq r$.

Now, given $r \in \mathbb{R}^+$, we denote by $\pi_0(\mathrm{VR}_r(Z))$ the quotient space $Z/\sim_r$, where two points $z, z' \in Z$ are such that $z \sim_r z'$ if and only if both z and z' lie in the same connected component from $\mathrm{VR}_r(Z)$. We write $[z]$ to refer to the class of z in $\pi_0(\mathrm{VR}_r(Z)) = Z/\sim_r$, where $r \in \mathbb{R}^+$ is implied by the context. The *0-homology group* $\mathrm{H}_0(\mathrm{VR}_r(Z))$ is the free $\mathbb{Z}_2$-vector space generated by the set $\pi_0(\mathrm{VR}_r(Z))$. The *0-persistent homology* of $\mathrm{VR}(Z)$, denoted as $\mathrm{PH}_0(Z)$, is given by the set of 0-homology groups $\left\{\mathrm{H}_0(\mathrm{VR}_r(Z))\right\}_{r\in\mathbb{R}^+}$ and the set of linear maps, $\left\{\rho^Z_{rs}\colon \mathrm{H}_0(\mathrm{VR}_r(Z)) \to \mathrm{H}_0(\mathrm{VR}_s(Z))\right\}_{r\leq s}$ that are induced by the inclusions $\mathrm{VR}_r(Z) \subseteq \mathrm{VR}_s(Z)$ for all $r \leq s$.

The 0-persistent homology is an example of a *persistence module*. Formally, a persistence module V is a set of free finite $\mathbb{Z}_2$-vector spaces $\{V_r\}_{r\in\mathbb{R}^+}$ together with a set of linear maps $\{\rho^V_{rs}\colon V_r \to V_s\}_{r\leq s}$ called *structure maps*. Another example of persistence module is the *interval module*, κ^a, for $a > 0$, where $\kappa^a_r = \mathbb{Z}_2$ for all $r < a$ and $\kappa^a_r = 0$ otherwise, with structure maps $\{\rho^{\kappa^a}_{rs}\colon \kappa^a_r \to \kappa^a_s\}_{r\leq s}$ being the identity map whenever $s \leq a$ and the zero map otherwise. In addition, we define κ^0 to be such that $\kappa^0_0 = \mathbb{Z}_2$ and $\kappa^0_r = 0$ for all $r > 0$. Also, we denote by κ^∞ the persistence module that consists of $\kappa^\infty_r = \mathbb{Z}_2$ for all $r \in \mathbb{R}^+$; with structure maps $\{\rho^{\kappa^\infty}_{rs} : \kappa^\infty_r \to \kappa^\infty_s\}_{r\leq s}$ being the identity maps. Finally, a *persistence morphism* $f\colon V \to U$ between persistence modules V and U is a set of linear maps $\big\{f_r\colon V_r \to U_r\big\}_{r\in\mathbb{R}^+}$ that commute with the structure maps ρ^V of V and ρ^U of U. If f_r is an isomorphism for all $r \in \mathbb{R}^+$, then f is called a *persistence isomorphism.* An example of a persistence morphism is $\kappa^a \to \kappa^b$, for $a \geq b$, where $\kappa^a_r \to \kappa^b_r$ consists of the identity map whenever $b \leq r \leq a$ and the zero map otherwise. An example of persistence isomorphism is $\kappa^\infty \to \kappa^\infty$ where $\kappa^\infty_r \to \kappa^\infty_r$ consists of the identity map for all $r \in \mathbb{R}^+$.

1.1 Connected Components, Barcodes and Triplet Merge Trees

Intuitively, $\mathrm{PH}_0(Z)$ encapsulates the evolution of connected components in $\mathrm{VR}(Z)$. This way, all classes in $\mathrm{PH}_0(Z)$ are *born* at 0 since $\pi_0(\mathrm{VR}_0(Z)) = \big\{[z_1],[z_2],\dots,[z_n]\big\}$. As the filtration parameter increases, $\mathrm{PH}_0(Z)$ records the death values of such classes. Specifically, a class $[z_j] \in \pi_0(\mathrm{VR}_0(Z))$ is said to *die at value* $b > 0$ if:

1) $\rho^Z_{0\ell}([z_j]) = [z_j]$ for all $\ell \in \mathbb{R}^+$ with $\ell < b$.
2) $\rho^Z_{0b}([z_j]) = [z_i]$, for some $i < j$. That is, $[z_j] + [z_i] \in \ker \rho^Z_{0b}$.

In addition, we say that $[z_j] \in \pi_0(\mathrm{VR}_0(Z))$ *dies at* 0 whenever there exists $i < j$ such that $d_Z(z_i, z_j) = 0$. Finally, observe that, the component $[z_1]$ never dies.

A way to track the evolution of connected components is via *triplet merge trees* [12]:

$$\mathrm{TMT}(Z) = \big\{\,\mathrm{triplet}(j)\colon j \in \{2,3,\dots,n\}\big\} \subset Z \times \mathbb{R}^+ \times Z\,\big\},$$

where $\mathrm{triplet}(j) = (z_j, b_j, z_i)$ is such that $[z_j] \in \pi_0(\mathrm{VR}_0(Z))$ dies at value $b_j \geq 0$ and $\rho^Z_{0b_j}([z_j]) = [z_i]$, where $i \in \{1,2,\dots,j-1\}$ is as small as possible. Using $\mathrm{TMT}(Z)$, we relate the elements of $\pi_0(\mathrm{VR}_0(Z))$ to death values of connected components. It is worth mentioning that the death values do not depend on the order chosen for the points of Z.

To store such death values, we use *barcodes.* The barcode $\mathrm{B}(Z) = (S^Z, m^Z)$ is a multiset where S^Z is the set of death values $b > 0$, and $m^Z(b)$ is the number of generators of $\mathrm{H}_0(\mathrm{VR}_0(Z))$ that die at b, for a fixed $b > 0$. The *representation of* $\mathrm{B}(Z)$ is $\mathrm{Rep}\,\mathrm{B}(Z) = \big\{\ (b,\ell)\ \mid\ b \in S^Z \text{ and } \ell \in \{1,2,\dots,m^Z(b)\}\ \big\}$. The following is a well-known fact (see Theorem 1.2 of [2]) that we have adapted to our case.

▶ **Lemma 1.1.** *There is a persistence isomorphism*

$$\mathrm{PH}_0(Z) \xrightarrow{\ \simeq\ } \Big(\bigoplus_{b\in\mathbb{R}^+} \bigoplus_{\ell\in\{1,2,\dots,m^Z(b)\}} \kappa^b\Big) \oplus \kappa^\infty.$$

1.2 Induced Block Functions and 1-Lipschitz Maps

To introduce the notion of block functions that will be used later to define induced matching distances, we need the operators $\ker^\pm_b$ defined as follows. For all $b > 0$,

$$\ker^-_b(Z) = \bigcup_{0\leq r<b} \ker(\rho^Z_{0r}) = \big\langle\big\{[z_j]+[z_i]\colon (z_j,b_j,z_i)\in\mathrm{TMT}(Z) \text{ and } b_j < b\big\}\big\rangle$$

and $\ker^+_b(Z) = \ker(\rho^Z_{0b}) = \big\langle\big\{[z_j]+[z_i]\colon (z_j,b_j,z_i)\in\mathrm{TMT}(Z) \text{ and } b_j \leq b\big\}\big\rangle$.
We also define $\ker^-_0(Z) = 0$ and $\ker^+_0(Z) = \big\langle\big\{[z_j]+[z_i] \mid (z_j,0,z_i)\in\mathrm{TMT}(Z)\big\}\big\rangle$. Finally,

$$\ker^-_\infty(Z) = \bigcup_{0\leq r} \ker(\rho^Z_{0r}) \quad\text{and}\quad \ker^+_\infty(Z) = \mathrm{H}_0(\mathrm{VR}_0(Z))\,.$$

We consider now another set of n points $X = \{x_1, x_2, \dots, x_n\}$ together with a symmetric non-negative function $d_X : X \times X \to \mathbb{R}^+$, and a bijection $f_\bullet : X \to Z$ mapping x_i to z_i for all $i \in \{1, 2, \dots, n\}$. Such a bijection induces an isomorphism $f_0 \colon \mathrm{H}_0(\mathrm{VR}_0(X)) \to \mathrm{H}_0(\mathrm{VR}_0(Z))$. In particular, we have that $f_0(\ker_0^-(X)) = 0$, and also

$$f_0\big(\ker_a^-(X)\big) = \big\{[f_0(x_j)] + [f_0(x_i)] \colon (x_j, a_j, x_i) \in \mathrm{TMT}(X) \text{ and } a_j < a\big\} \text{ for all } a > 0, \text{ and}$$
$$f_0\big(\ker_a^+(X)\big) = \big\{[f_0(x_j)] + [f_0(x_i)] \colon (x_j, a_j, x_i) \in \mathrm{TMT}(X) \text{ and } a_j \le a\big\} \text{ for all } a \in \mathbb{R}^+.$$

The *induced block function* $\mathcal{M}_f^0 \colon \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{N}$ is defined, for all $a, b \in \mathbb{R}^+$, by

$$\mathcal{M}_f^0(a,b) = \dim\left(\frac{f_0(\ker_a^+(X)) \cap \ker_b^+(Z)}{f_0(\ker_a^-(X)) \cap \ker_b^+(Z) + f_0(\ker_a^+(X)) \cap \ker_b^-(Z)}\right)$$

Intuitively, $\mathcal{M}_f^0(a,b)$ is the amount of connected components of $\mathrm{H}_0(\mathrm{VR}_0(X))$ that die at a and are sent by f_0 to connected components of $\mathrm{H}_0(\mathrm{VR}_0(Z))$ that die at b. It is well-defined because $f_0(\ker_a^\pm(X)) \subseteq \mathrm{H}_0(\mathrm{VR}_0(Z))$ and $\ker_b^-(Z) \subseteq \ker_b^+(Z) \subseteq \mathrm{H}_0(\mathrm{VR}_0(Z))$. This definition is an adaptation, to our setting, of the one given in [6], whose worst-case time complexity is $O(n^3)$ as stated in [14]. Unlike in [6], it is possible that $\mathcal{M}_f^0(a,b) \neq 0$ for a pair (a,b) with $a < b$, and we cannot guarantee the very good properties presented in [6] such as, for example, linearity as, in general, $f_\bullet$ does not induce a persistence morphism $f \colon \mathrm{PH}_0(X) \to \mathrm{PH}_0(Z)$.

However, there is a simple workaround. Given $\delta \in \mathbb{R}^+$, we define X^δ to be the set X together with the symmetric non-negative function $d_{X^\delta} : X \times X \to \mathbb{R}^+$ given by $d_X^\delta(x, x') = d_X(x, x') + \delta$. Now, for $\delta \in \mathbb{R}^+$ big enough, $f_\bullet^\delta : X^\delta \to Z$ is a 1-*Lipschitz map*, meaning that $d_{X^\delta}(x, x') \ge d_Z(f_\bullet^\delta(x), f_\bullet^\delta(x'))$ for all $x, x' \in X^\delta$. Taking advantage of such property, adapting Theorem 5.3 from [10] and Theorem 5.1 from [6], we derive the following result.

▶ **Lemma 1.2.** *If $\delta \in \mathbb{R}^+$ is big enough, then $f^\delta \colon \mathrm{PH}_0(X^\delta) \to \mathrm{PH}_0(Z)$ is a persistence morphism and we have that*

$$f^\delta \simeq \left(\bigoplus\nolimits_{b \in \mathbb{R}^+} \bigoplus\nolimits_{a \ge b} \bigoplus\nolimits_{i \in \{1,2,\dots,\mathcal{M}^0_{f^\delta}(a,b)\}} (\kappa^a \to \kappa^b)\right) \oplus (\kappa^\infty \to \kappa^\infty).$$

1.3 Induced Matching Distance

Given barcodes (S, m) and (S', m'), a *matching* is a bijection $\sigma \colon \mathrm{Rep}\,(S, m) \to \mathrm{Rep}\,(S', m')$. To define the induced matching $\sigma_f^0 \colon \mathrm{Rep}\, B(X) \to \mathrm{Rep}\, B(Z)$, we first consider $\delta \in \mathbb{R}^+$ big enough so that $f_\bullet^\delta : X^\delta \to Z$ is 1-Lipschitz. Now, a consequence of Lemma 1.1 and Lemma 1.2 is that we can always define a matching $\sigma_1 \colon \mathrm{Rep}\, B(X^\delta) \to \mathrm{Rep}\, B(Z)$. Second, from the equality $\ker_a^\pm(X) = \ker_{a+\delta}^\pm(X^\delta)$ for all $a \in \mathbb{R}^+$, we have

$$\phi \colon S^X \xrightarrow{\simeq} S^{X^\delta} \text{ is such that } \phi(a) = a + \delta, \text{ and } m^X(a) = m^{X^\delta}(a + \delta), \text{ for all } a \in S^X.$$

In particular, we have another matching $\sigma_2 \colon \mathrm{Rep}\, B(X) \to \mathrm{Rep}\, B(X^\delta)$. Altogether, since $\mathcal{M}_f^0(a,b) = \mathcal{M}_{f^\delta}^0(a + \delta, b)$, we define a matching $\sigma_f^0 \colon \mathrm{Rep}\, B(X) \to \mathrm{Rep}\, B(Z)$ as $\sigma_1 \circ \sigma_2$.

We now introduce the novel concept of *matching distance induced* by bijections between point sets endowed with symmetric non-negative functions. It is worth mentioning papers [14, 13], where the related concept of matching diagrams induced by set mappings between finite metric spaces was defined. Fixed $q \in \mathbb{N}$, the *induced matching distance* is the symmetric non-negative function $d^q_{f_0}$ defined as:

$$d^q_{f_0}(\mathrm{B}(X), \mathrm{B}(Z)) = \left(\sum_{\substack{(a,\ell) \in \operatorname{Rep} B(X) \\ \sigma^0_f((a,\ell)) = (b,\ell')}} |a-b|^q \right)^{1/q} = \left(\sum_{a,b \in \mathbb{R}^+} \mathcal{M}^0_f(a,b) \cdot |a-b|^q \right)^{1/q} \quad (1)$$

Observe that the q-Wasserstein distance [4] is always bounded by the induced matching distance. The paper's main result states that the induced matching distance is continuous.

▶ **Theorem 1.3.** *Fixed $\varepsilon > 0$, consider a bijection $f_\bullet \colon X \to Z$ such that $|d_X(x,y) - d_Z(f_\bullet(x), f_\bullet(y))| < \varepsilon$ for all $x, y \in X$. Then, $d^q_{f_0}(B(X), B(Z)) \leq (n-1)^{1/q} \cdot \varepsilon$.*

Proof. To start, if $|a-b| > \varepsilon$, then $\mathcal{M}^0_f(a,b) = 0$, since either $f_0(\ker^+_a(X)) \subseteq \ker^-_b(Z)$ or $\ker^+_b(Z) \subseteq f_0(\ker^-_a(X))$ holds. On the other hand, recall that, $\sum_{b \in \mathbb{R}^+} \mathcal{M}^0_f(a,b) = m^X(a)$ and also $n - 1 = \sum_{a \in \mathbb{R}^+} m^X(a)$. Altogether, since $\mathcal{M}^0_f(a,b) \neq 0$ if and only if $|a-b| < \varepsilon$, then $(n-1)^{1/q}\varepsilon$ is an upper bound for (1). ◀

We conclude this section by illustrating why, in certain cases, it is preferable to compare barcodes using the induced matching distance rather than the q-Wasserstein distance.

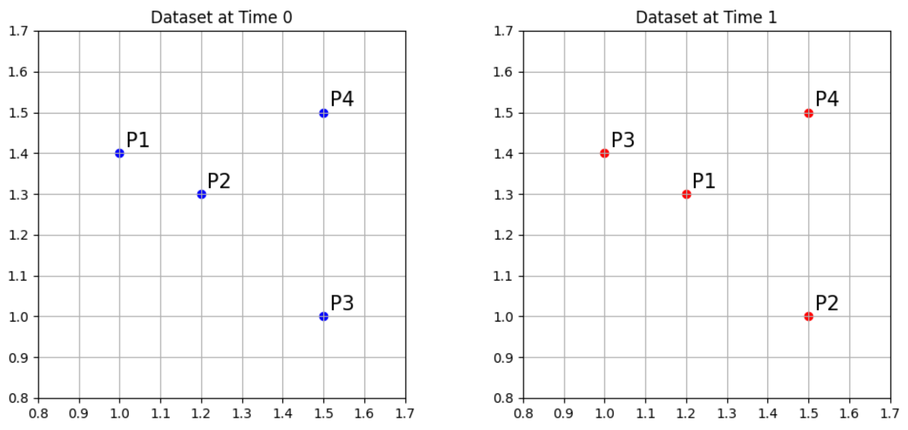


Figure 1 Point clouds X_0 (left) and X_1 (right) with the same points but different labels.

Consider the point clouds X_0 and X_1 pictured in Fig. 1. Observe that they have the same shape, and therefore, their barcodes are equal, not reflecting that the points in the clouds have changed their labels. We aim to preserve the information conveyed by labels since points may represent the states of distinct agents, and the labels identify the agents.

In Fig. 2 (left), we can see the matching that produces a q-Wasserstein distance of 0. In Fig. 2 (right), we can see the matching induced by the bijection $X_0 \to X_1$ resulting from the label changes, produces a nonzero induced matching distance, which is more coherent.

Finally, recall that, by definition, to compute the Wassertein distance, we have to find the matching that achieves the minimal sum of the differences of the machined deaths, whereas the given bijection between points immediately provides the matching that produces the induced matching distance. Therefore, we can think of the induced matching distance as an alternative way to compare diagrams effectively.

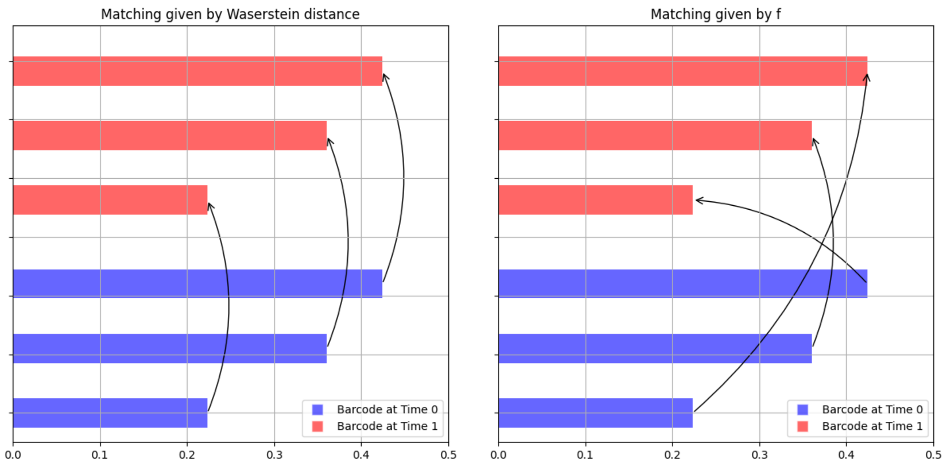


Figure 2 The matching that produces the q-Wasserstein distance (left) and the matching that produces the induced matching distance (right) between $\mathrm{B}(X_0)$ and $\mathrm{B}(X_1)$. The arrows indicate the barcodes of $\mathrm{B}(X_0)$ matched with the ones of $\mathrm{B}(X_1)$. As one can observe, the matching induced by the bijection takes into account the change of labels in Fig.1.

2 Robot Fleet Navigation Analysis via the Induced Matching Distance

As an application, we aim to compare, via the induced matching distance, three local navigation algorithms or *behaviors* for robots: human like (HL) [8], optimal reciprocal collision avoidance (ORCA) [15], and social force (SF) [9]. Simulated robots use these algorithms to reach their targets, taking control actions operating on the local environment around them, and also record their trajectories, composed of poses and orientations $(x, y, \alpha) \in \mathbb{R}^2 \times [0, 2\pi)$.

For our experiments, we use the Navground social navigation simulator [7]. Specifically, we consider simulations in a scenario that represents a corridor with a length of 15 m and a width of 3.5 m, having both ends connected. In each simulation, 10 autonomous robots apply the same behavior to navigate the corridor, with 5 robots moving to the left and 5 moving to the right. These robots mimic smart wheelchairs with differential drive kinematics, radius 0.4 m, target speed 1.2 m s^{-1}, and maximal speed 1.66 m s^{-1}. At the beginning of the simulations, the robots are randomly distributed in the corridor and attempt to follow their assigned direction while avoiding collisions. We collect the pose and orientation of each robot every 0.1 s for 90 s, resulting in a set of 10 3-dimensional time series, each containing 900 values. Fig. 3 shows the trajectories followed by the 10 robots for 3 different simulations.

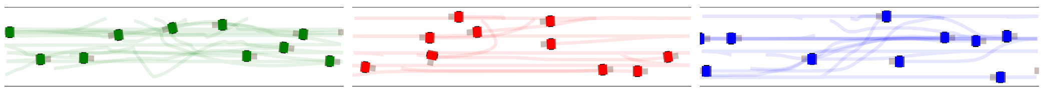


Figure 3 The first 10 seconds of the trajectories followed by the robots in the corridor scenario are depicted, corresponding to the behaviors SF (left), ORCA (center), and HL (right).

In each simulation, the i-th robot is associated with a 3-dimensional time series a^i representing its trajectory, being $a^i = \left\{a^i_t\right\}_{t=1}^{900}$ and $a^i_t = (x^i_t, y^i_t, \alpha^i_t)$. Now, for $t = 1, 2, \ldots, 850$, we obtain the set $Z_t = \left\{z^i_t\right\}_{i=1}^{10}$, where $z^i_t = \{a^i_t, a^i_{t+10}, \ldots, a^i_{t+50}\} \subset a^i$ is a time window. We use dynamic time warping as the symmetric non-negative function to build a sequence of Vietoris-Rips filtrations $\left\{\mathrm{VR}_0(Z_t)\right\}_{t=1}^{850}$. Dynamic time warping is a symmetric non-negative measure that enables non-linear alignment between two time series by finding an optimal warping path that minimizes the cumulative distance between corresponding points [11, 1]. For $t = 1, 2, \dots, 800$, we compute the induced block function $\mathcal{M}^0_{f^t}$ from the bijection $f^t_\bullet : Z_t \to Z_{t+50}$, resulting in a sequence of induced matching distances $\{d^1_{f^t_0}(B(Z_t), B(Z_{t+50}))\}_{t=1}^{800}$ that is called the *induced matching signal.*

The induced matching signal provides an intuition on the stability of the trajectories. Ideally, the robots tend to organize themselves and end up forming lanes in the corridor, moving forward indefinitely without ever correcting their trajectories or changing their speed. This is called a stable state. When a simulation has stabilized, the set $Z_t = \{z_t^1, z_t^2, \dots, z_t^{10}\}$ may change as t changes, but the distances between its elements given by dynamic time warping remain similar, so the Vietoris-Rips filtrations are also similar and the induced matching distances are low. A decreasing induced matching signal indicates that the corresponding simulation is stabilizing, with the robots becoming better organized.

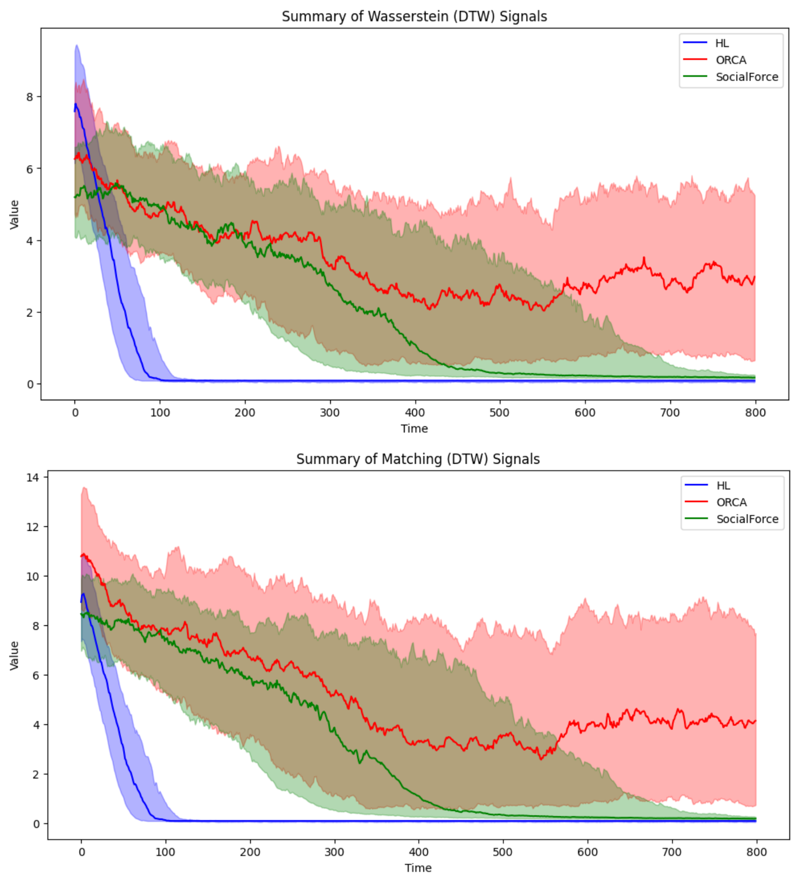


Figure 4 Wasserstein and induced matching signals for the 600 simulations in the corridor scenario. Bold lines show the median signal for each behavior, and shaded bands represent the interquartile ranges.

We have performed 200 simulations for each behavior, getting 600 Wasserstein and induced matching signals summarized in Fig. 4. At first glance, the signals vary significantly depending on the behavior. HL and SF tend to stabilize the simulations in less than 90 s, although at different speeds, while ORCA generally fails to stabilize within that time.

To show, in a quantitative way, that the three behaviors can be well distinguished using our topology-based signal, we build a time series classifier based on neural networks. Following best practices [5], we trained a ResNet model [16] using Python [3, Chapter 8]. We split the set of 600 signals into 420 for training, 60 for validation, and 120 for testing. After 100 training epochs, we got a ResNet model whose confusion matrices are shown in Fig. 5. The model demonstrates perfect performance on the training set and achieves an accuracy and a macro-average F1-score of 97.5% on the test set, indicating that the induced matching signals effectively distinguish between the different navigation behaviors.

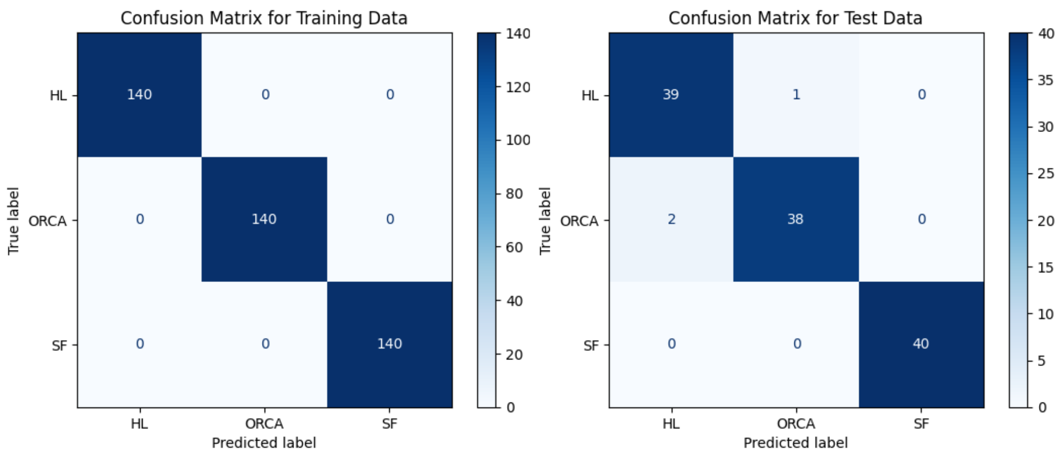


Figure 5 The confusion matrices for the ResNet classifier on the induced matching signals.

3 Conclusions

We introduced a topology-based tool for comparing discrete structures represented by symmetric non-negative functions and applied it to monitor the dynamic behavior of trajectory groups. The study of its stability and extension to higher dimensions remains future work.
Code availability: The source code for the experiments presented in this paper can be found in https://github.com/Cimagroup/induced-matching-distance-navground.
Acknowledgements: Research partially supported by REXASI-PRO H-EU project, Grant agreement ID: 101070028. Torras-Casas research has been partially funded by Ministerio de Ciencia e Innovación project TED2021-129438B-I00 and the French Agence Nationale de la Recherche project ANR-22-CPJ1-0047-01.

References

1 D.J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *Int. Conf. on Knowledge Discovery and Data Mining*, pages 359–370, 1994.
2 M.B. Botnan and W. Crawley-Boevey. Decomposition of persistence modules. *Proceedings of the American Mathematical Society*, 148:4581–4596, 2020. `doi:10.1090/proc/14790`.
3 V. Cerqueira and L. Roque. *Deep Learning for Time Series Cookbook: Use PyTorch and Python recipes for forecasting, classification, and anomaly detection.* Packt Ltd., 2024.
4 D. Cohen-Steiner, H. Edelsbrunner, J. Harer, and Y. Mileyko. Lipschitz functions have lp-stable persistence. *Found. Comput. Math.*, 10(2):127–139, 2010. `doi:10.1007/s10208-010-9060-6`.
5 H.I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.A. Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

6 R. Gonzalez-Diaz, M. Soriano-Trigueros, and A. Torras-Casas. Partial matchings induced by morphisms between persistence modules. *Computational Geometry*, 112:101985, 2023. doi:10.1016/j.comgeo.2023.101985.

7 J. Guzzi. Navground (navigation playground). URL: https://github.com/idsia-robotics/navground.

8 J. Guzzi, A. Giusti, L.M. Gambardella, G. Theraulaz, and G.A. di Caro. Human-friendly robot navigation in dynamic environments. In *Int. Conf. Robot. Autom.*, 2013. doi:10.1109/ICRA.2013.6630610.

9 D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical Review E*, 51, 05 1998. doi:10.1103/PhysRevE.51.4282.

10 E. Jacquard, V. Nanda, and U. Tillmann. The space of barcode bases for persistence modules. *J. Appl. Comput. Topol.*, 7:1–30, 2023. doi:10.1007/s41468-022-00094-6.

11 H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoust. Speech Signal Process.*, 26(1):43–49, 1978. doi:10.1109/TASSP.1978.1163055.

12 D. Smirnov and D. Morozov. Triplet merge trees. In *Topological Methods in Data Analysis and Visualization V*, pages 19–36. Springer, 2020.

13 A. Torras-Casas and R. Gonzalez-Diaz. Properties and stability of persistence matching diagrams, 2024. arXiv:2409.14954.

14 A. Torras-Casas, E. Paluzo-Hidalgo, and R. Gonzalez-Diaz. Topological quality of subsets via persistence matching diagrams, 2024. arXiv:2306.02411.

15 J. van den Berg, S.J. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *Robotics Research*, pages 3–19, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

16 Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *Int. Joint Conf. Neural Netw.*, pages 1578–1585, 2017.

Minimum spanning blob-trees

Katharina Klost[1], Marc van Kreveld[2], Daniel Perz[3], Günter Rote[1], and Josef Tkadlec[4]

1 **Freie Universität Berlin, Institut für Informatik**
`{kathklost,rote}@inf.fu-berlin.de`
2 **Department of Information and Computing Sciences, Utrecht University**
`m.j.vankreveld@uu.nl`
3 **University of Perugia**
`daniel.perz@unipg.it`
4 **Charles University**
`josef.tkadlec@iuuk.mff.cuni.cz`

Abstract

We investigate blob-trees, a new way of connecting a set of points, by a mixture of enclosing them by cycles (as in the convex hull) and connecting them by edges (as in a spanning tree). We show that a minimum-cost blob-tree for n points can be computed in $O(n^3)$ time.

1 Introduction

Any introductory course on computational geometry will treat convex hulls and minimum spanning trees (MSTs) for a set P of n points in the plane. These are the least-cost structures that enclose or connect the points. In this paper we investigate a new structure, the *blob-tree*, that combines the ideas of enclosing and connecting. A *blob* is a simple polygon, and all points of P enclosed in a blob are considered to be connected to each other. A *tree-edge* is a segment between two points of P, and it connects its two endpoints. A blob-tree is a collection of blobs and tree-edges that collectively connects all points of P, see Figure 1. The convex hull and the MST of P are special cases of a blob-tree.

We are looking for the blob-tree that minimizes the total length of all edges drawn: the perimeter of the blobs plus the tree-edges. As we show in Lemma 1 below, the blobs are convex and disjoint. Morever, when contracting the blobs to vertices, the tree-edges form a tree, so the name "blob-tree" is warranted.

While blob-trees appear interesting in their own right, similar structures have been used in information visualization, in particular in KelpFusion [8]. This visualization style is based

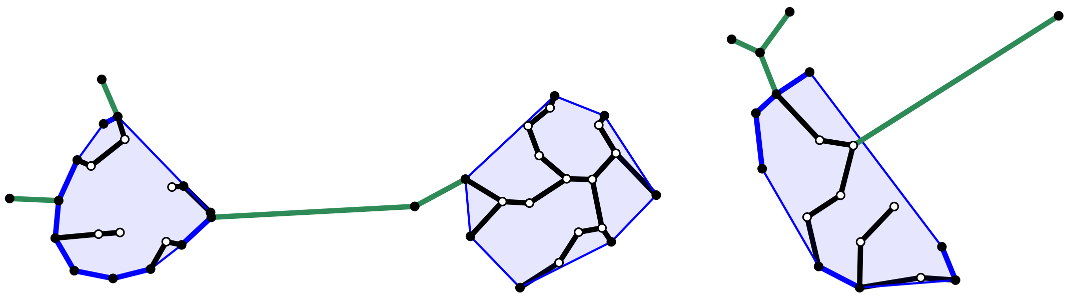
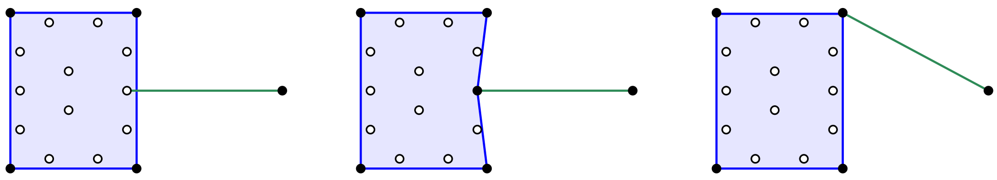


Figure 1 The optimal blob-trees on two point sets (computed by our implementation [4]). Blob edges are blue, tree-edges are green. The cost of the solution is the total length of blue and green. MST edges are drawn heavier.



(i): crossing allowed
blob perimeter: 58
edge length: 16
total cost: 74

(ii): no crossings
blob perimeter: ≈ 58.12
edge length: 16
total cost: ≈ 74.12

(iii): no crossings & convex
blob perimeter: 58
edge length: 17
total cost: 75

Figure 2 Three variants of blob-trees for the same point set, and their costs.

on *shortest-path graphs* [5], a one-parameter family of connected structures whose extremes are the convex hull and the MST, without regard to a specific optimization criterion. Other related work considers placing shortest fences for subsets of points [1, 2, 3].

As seen in Figures 1 and 2, an optimal solution might involve a tree-edge that crosses a boundary of a blob. If we disallow such crossings (version (ii)), the optimal solution could get longer and might include non-convex blobs. If we further required the blobs to be convex (version (iii)), the optimal solution might become even longer, see Figure 2.

In this paper, we concentrate on the original problem. It turns out that the optimal solution can be obtained from the MST by replacing several of its subtrees by blobs (Lemma 2); the other two variants (ii) and (iii) do not have this property. As our main result we show that the optimal blob-tree can be computed in $O(n^3)$ time, and we implement the algorithm in Python [4]. Efficient computation of optimal blob-trees in variants (ii) or (iii) remains open.

Our algorithm is based on dynamic programming and builds upon ideas from [6]. That paper shows how to compute a single optimal convex polygon (or blob) in a point set that has certain properties, using dynamic programming. For example, it can compute the smallest area or perimeter convex polygon with k points on the boundary, or the smallest area convex polygon containing at least k points inside or on the boundary in $O(kn^3)$ time. We show how to deal with multiple blobs and with the minimum spanning connecting structure to compute an overall minimum solution. All proofs that are not given here are in the full version [7].

2 Preliminaries

We assume that the point set P is in general position. In particular, we assume that the pairwise distances between points are unique and no three points are on a line. This implies that the MST of the point set is unique. We also assume that no two points have the same x- or y-coordinate.

▶ **Lemma 1.** *In any optimal solution: all blobs are convex polygons with vertices at P; any two blobs are disjoint; and when contracting the blobs to vertices, the tree-edges form a tree.*

Proof. The claims follow from the triangle inequality. First, we can replace any blob that does not have the claimed property with the convex hull of the points of P contained in it, creating a blob-tree with a strictly smaller total cost. Similarly, two intersecting blobs B_1 and B_2 can be replaced by the convex hull of the union $B_1 \cup B_2$.

Contraction of blobs creates an abstract graph G in which every blob and every singleton point that is not in a blob is represented by a vertex. For every tree-edge, G contains an edge between the corresponding vertices. Since the blob-tree is connected and spans all points,

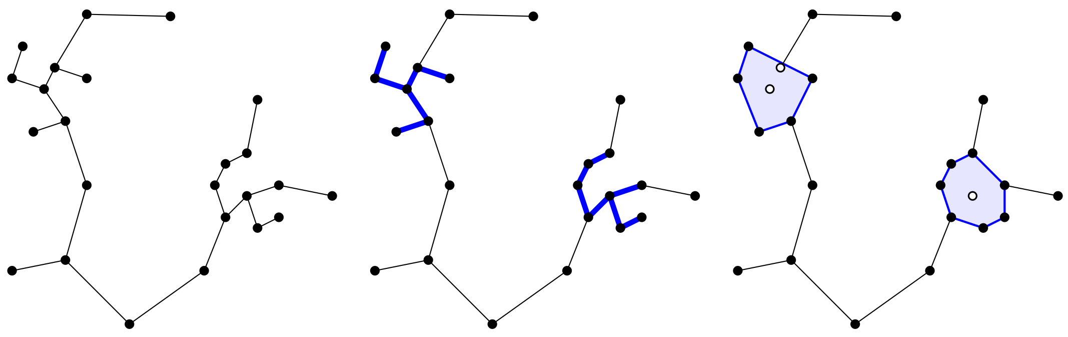


Figure 3 An MST, a potential subset of non-tree edges (fat blue), and the resulting blob-tree.

G is connected. If G contained a cycle, then any tree-edge corresponding to an edge from such a cycle could be removed from the blob-tree. So the graph G is a tree. ◀

Throughout the paper, T will denote the minimum spanning tree (MST) of P. We select the lowest point r as the root and direct all edges of T towards r. For a node u, we denote its parent in T by u' and the directed edge from u to its parent by uu'.

3 Structural insights

The following key lemma with its immediate corollary shows that the optimal solution can be obtained from the MST by replacing some of its subtrees by the respective blobs.

▶ **Lemma 2.** *Let T be the MST. Then, in any optimal blob-tree S, every blob is a convex hull of some subtree of T and every tree-edge is an edge of T.*

Lemmas 1 and 2 immediately yield the following corollary.

▶ **Corollary 3.** *Let B be a blob in any optimal blob-tree S. Let T be the MST. Then:*

(a) *A path in T that leaves B never comes back to B.*
(b) *The subgraph of T induced by the points in B is connected.*
(c) *No MST edge with both endpoints outside B intersects B.* ◀

The structure of optimal blob-trees can be used to develop a polynomial-time algorithm by dynamic programming. We extend the structure in two simple ways.

Firstly, we use the direction of all edges of T towards the root r. If we were to contract each blob to a single point, then we would have a normal rooted tree. Consequently, every blob that does not contain the root has one unique exit edge and every blob has zero or more entry edges. The exit from a blob towards the root can be via an edge from one of its hull vertices or via an edge from an interior blob point that crosses a hull edge. Similarly, blob entrances come in two types, see Figure 4.

Secondly, we consider a bottom-vertex triangulation of each blob, where the lowest point is connected to all other vertices on the boundary of the blob. We call the diagonals of the triangulation *chords* and the exterior edges *walls*. The chords and triangles serve to have simple (constant size) elements to be used to define substructures in the dynamic program. Optimal solutions to smaller subproblems are "transported" through a blob from any entrance to the unique exit, using the chords in between, to build solutions to larger subproblems.

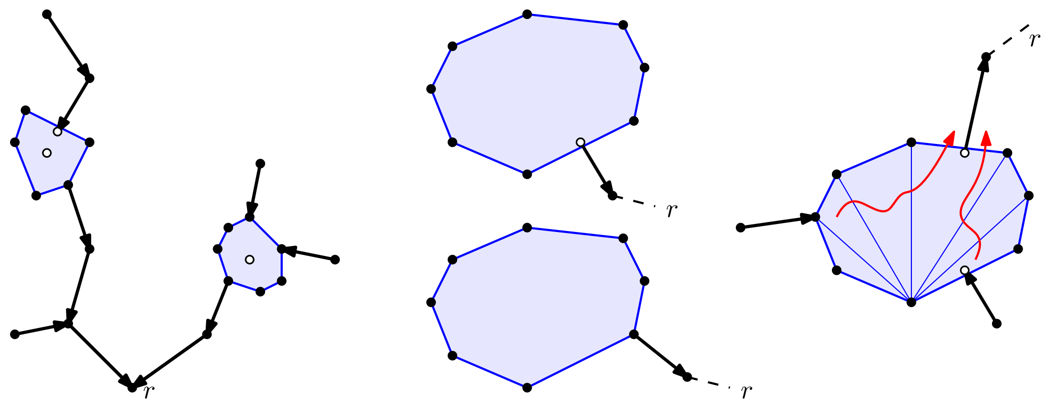


Figure 4 A rooted, directed blob tree, two types of exits, and a bottom-vertex triangulation of a blob with a dynamic programming order through it towards the root.

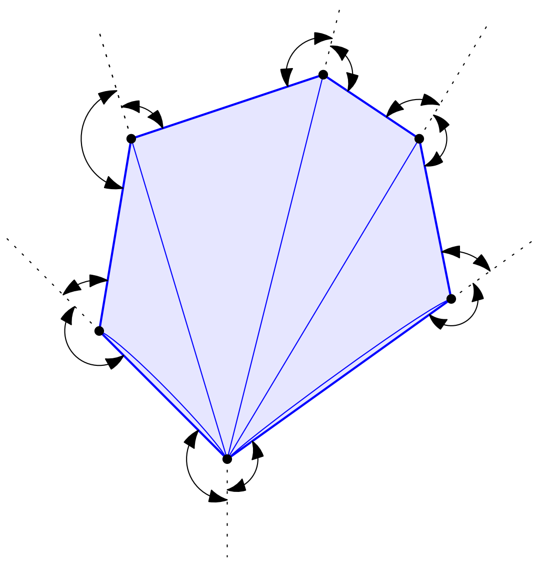


Figure 5 A blob is divided into triangles and digons.

There will be three types of subproblems: (i) edge subproblems, for each MST edge that is used as a blob-tree edge, (ii) chord subproblems, and (iii) wall subproblems. There are $n-1$ of the first type and $O(n^2)$ of the latter two types. Walls and chords do not have an implied orientation (unlike the edges of T). Each segment must be considered as a potential wall or chord separately in both orientations.

For uniformity, we draw chords from the bottom vertex of a blob also to its neighbors on the blob boundary, cutting off two degenerate *digons* from the blob, see Figure 5. Thus, a blob with k boundary points has $k-1$ chords and k walls, and it is cut into $k-2$ triangles, each bounded by two chords and one wall, and two digons, bounded by one chord and one wall.

Every chord has a *forward side*, towards the tree root, and a *backward side*; both are defined later in detail. Our algorithm proceeds from the leaves towards the root, as indicated by the red arrows in Figure 4. Thus, the subproblem associated to a chord consists of all points that are on the backward side, including the points that are in other blobs hanging

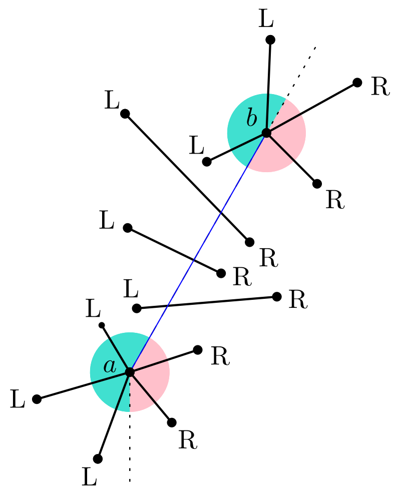


Figure 6 Labeling endpoints of MST-edges X_{ab}^{+} crossing ab or incident to a or b as left (L) and right (R). The neighbors of a and b are labeled right with respect to ab if the edges emanate in the pink region, otherwise they are left. (The shown edges cannot all be MST edges simultaneously.)

off the current blob in the blob tree. It turns out that the points that are involved in this subproblem can be uniquely identified with the help of the MST, without knowing the optimum blob tree.

Let a, b be a pair of points. Let X_{ab} be the set of edges of T that cross the line segment ab, and let X_{ab}^{+} denote the edges in X_{ab} plus the edges of T that are incident to a or b.

Sidedness. Assume that a is below b and consider the *boundary curve* that is obtained from the union of the vertical ray below a and the ray from a through b, see Figure 6. This curve subdivides the plane into a left and a right side. We call a point v that is a vertex of an edge in X_{ab} a *left* or *right endpoint*, depending on if it lies left or right of the boundary curve. The endpoints of edges in $X_{ab}^{+} \setminus X_{ab}$ that are not a or b are classified analogous.

▶ **Lemma 4.** *Every edge in X_{ab} has exactly one left and one right endpoint.*

Valid chords. Removing X_{ab}^{+} from T gives a forest. We call the segment ab a *valid chord* if no component in $T \setminus X_{ab}^{+}$ contains both a left and a right endpoint. For a valid chord, the components of $T \setminus X_{ab}^{+}$ can be partitioned into *left* and *right components*, depending on the characterization of the vertices from X_{ab}^{+} contained in the component. Note that this partitions $P \setminus \{a, b\}$.

The side assigned to the root r is the *forward* side of the chord and the other side the *backward* side. (If the lower point a of the chord is r, we arbitrarily declare the right side as the forward side. This convention simplifies the treatment of the "root blob".) If the forward side is the right side, then ab is a ***right-facing*** chord, otherwise it is a ***left-facing*** chord.

▶ **Lemma 5.** *All chords in an optimal solution are valid.*

Walls. Walls are the edges on the boundary of a blob in a counterclockwise traversal. Let b, c be a pair of points. Then $uu' \in X_{bc}$ is an ***entry edge*** for $\overrightarrow{bc}$ if it crosses $\overrightarrow{bc}$ from right to left (from outside the blob to inside) and an ***exit edge*** otherwise.

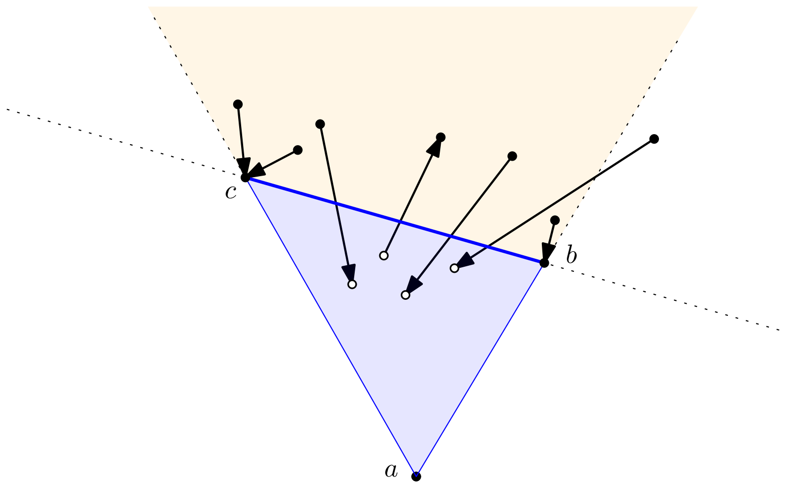


Figure 7 Six entry edges and one exit edge for the triangle abc. The four edges that cross bc are associated to the wall bc. sd

Triangles and Digons. Among the non-root triangles in to which a blob is decomposed from the lowest vertex, there is a unique wall, where the edge towards the root exits. If this wall is part of a triangle, it is an *exit triangle* the other triangles are LR-triangles (left-to-right), RL-triangles (right-to-left), depending on the direction in which the root lies, see Figure 8. Points adjacent to vertices of the blob are assigned to the triangles or digons according to the extension of the triangles by the rays through a and the other two vertices.

For every triangle abc with lowest point a, we can, by analyzing the tree edges that cross the sides or are incident to the vertices a, b, c, classify it as a potential LR-triangle, RL-triangle or exit triangle. A similar classification is obtained for digons. The details are given in [7, Appendix D].

The following lemma shows that this concept of valid triangles and digons agrees with the intended meaning:

▶ **Lemma 6.** *Let B be a blob in an optimal solution. Then,*

(a) There is exactly one exit triangle or digon.
(b) Every other triangle and digon in the decomposition of the blob from the lowest point is a valid LR-triangle, RL-triangle or a valid entry digon.

4 The algorithm

The algorithm does not actually check that the blobs that we form are convex. The only property that we implicitly enforce is that each blob is star-shaped around the lowest vertex a. Thus, the algorithm considers also nonconvex blobs for potential solutions. However, we know from Lemma 1 that such solutions cannot be optimal. As mentioned, we consider three type of subproblems.

Edge problems. For each MST edge uu', we define a problem `edge`$[u]$, for the optimum solution in which uu' is used as a tree-edge. In addition, for the root r, we have `edge`$[r]$ for the overall problem. We define V_u as the set of points in the subtree of T rooted at u. We define the *size* of the problem as the cardinality of V_u.

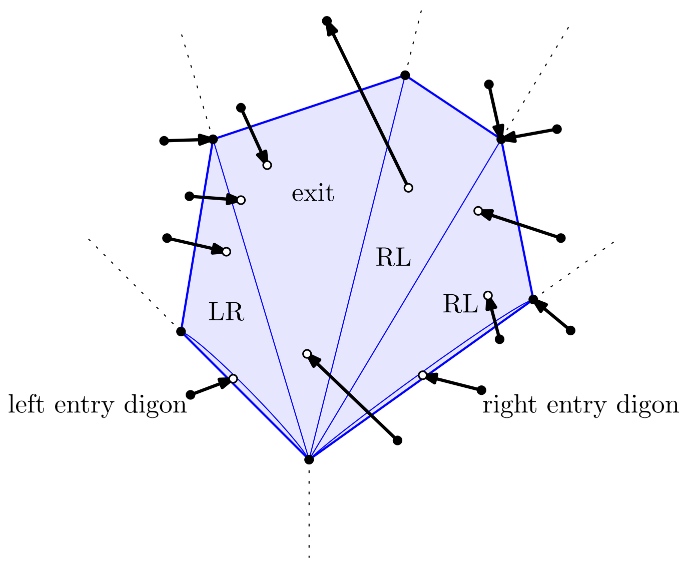


Figure 8 An example with one exit triangle, several LR- and RL-triangles, and two entry digons.

Chord problems. For each valid chord ab, we define a problem $\texttt{chord}[a, b]$ for the optimum solution on the backward side of ab, supposing that ab occurs as a chord of a blob in the solution. We define V_{ab} as the set of all points that are in a backward component of $T \setminus X^+_{ab}$. We define the *size* of the problem $\texttt{chord}[a, b]$ as the cardinality of V_{ab}. For clarity, we will sometimes write V_{ab} as $\overleftarrow{V}_{ab}$ for a right-facing chord and as $\overrightarrow{V}_{ab}$ for a left-facing chord.

Wall problems. We denote by W_{abc} the set of endpoint of entry edges for abc that lie outside of the blob. For each pair of points b, c, we denote by W_{bc} the set of points w where $ww' \in X_{bc}$ is an entry edge for bc. We will frequently need the values $\sum_{u \in W_{abc}} \texttt{edge}[u]$. We split this into

$$\sum_{u \in W_{abc}} \texttt{edge}[u] = \sum_{u \in W_{bc}} \texttt{edge}[u] + \sum_{u \in W_{abc},\ uu' \text{ entry edge into } b \text{ or } c} \texttt{edge}[u]$$

To speed up the evaluation of this term, the first sum is stored as the solution of an auxiliary subproblem, $\texttt{wall}[bc]$; the second sum can be computed in constant time, because there are only a constant number of entry edges incident to b and c, because the degree of a Euclidean MST is bounded by 6.

The size of this problem is defined as the sum of the sizes of the problems $\texttt{edge}[u]$. If bc is a wall in an optimal solution, the subtrees rooted at the nodes $w \in W_{bc}$ are disjoint. However, we don't check this condition. Hence the size can be bigger than n.

Relation between subproblems. In [7, Appendix G, Lemmas 9 and 10], we describe the relations between the sets of points that define the subproblems. These lemmas ensure that the solution of every subproblem depends only on problems of smaller size.

Preprocessing. In a preprocessing phase, we determine the size of each subproblem, in order to know the order in which we have to solve the subproblems. In [7, Appendix H, Lemma 11], we give details about the extra information gathered in the preprocessing step.

Chord problems. For each valid chord ab, we determine its sidedness. If it is right-facing, we consider all possibilities for an LR-triangle abc, as well as the possibility that ab is a left entering digon. Left-facing chords are analogous. More details are in [7, Appendix I].

Edge problems. For each MST edge uu', we have two possibilities. If u is not in a blob, all incoming MST edges of u must be tree-edges, and we can accumulate the values of the corresponding subproblems. If u is in a blob, we consider all potential exit triangles abc for which the uu' is the exiting edge crossing the wall bc, as well as the analogous possibility of an exit digon. Details are given in [7, Appendix I].

▶ **Theorem 7.** *The dynamic program above correctly solves the minimum blob-tree problem in $O(n^3)$ time.*

Proof. For each size, we first solve the `edge` problems, then the `wall` problems, and then the `chord` problems. Lemmas 9 and 10 in [7, Appendix G] show that every subproblem solution that is needed is already computed, and correctness follows from these lemmas.

The straightforward running time analysis is given in [7, Appendix J]. ◀

Acknowledgments. We thank Bettina Speckmann for co-proposing this problem, and Philipp Kindermann for hosting GG Week 2024 in Trier, where this research started.

References

1 Mikkel Abrahamsen, Panos Giannopoulos, Maarten Löffler, and Günter Rote. Geometric multicut: shortest fences for separating groups of objects in the plane. *Discrete Comput. Geom.*, 64:575–607, 2020. doi:10.1007/s00454-020-00232-w.

2 Esther M. Arkin, Samir Khuller, and Joseph S. B. Mitchell. Geometric knapsack problems. *Algorithmica*, 10(5):399–427, 1993. doi:10.1007/BF01769706.

3 Therese Biedl, Éric Colin de Verdière, Fabrizio Frati, Anna Lubiw, and Günter Rote. Finding a shortest curve that separates few objects from many. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry (SoCG 2025)*, volume 332 of *Leibniz International Proceedings in Informatics (LIPIcs)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2025. To appear.

4 Blobtrees. Github repository. URL: https://github.com/guenterrote/blobtrees.

5 Mark de Berg, Wouter Meulemans, and Bettina Speckmann. Delineating imprecise regions via shortest-path graphs. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 271–280, 2011. doi:10.1145/2093973.2094010.

6 David Eppstein, Mark Overmars, Günter Rote, and Gerhard Woeginger. Finding minimum area k-gons. *Discrete Comp. Geom.*, 7:45–58, 1992. doi:10.1007/BF02187823.

7 Katharina Klost, Marc van Kreveld, Daniel Perz, Günter Rote, and Josef Tkadlec. Minimum spanning blob-trees, 2025. arXiv:2503.02439.

8 Wouter Meulemans, Nathalie Henry Riche, Bettina Speckmann, Basak Alper, and Tim Dwyer. Kelpfusion: A hybrid set visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 19(11):1846–1858, 2013. doi:10.1109/TVCG.2013.76.

Defective Linear Layouts of Graphs[*]

Michael A. Bekos[1], Carla Binucci[2], Emilio Di Giacomo[2], Walter Didimo[2], Luca Grilli[2], Maria Eleni Pavlidi[1], Alessandra Tappini[2], and Alexandra Weinberger[3]

1 **University of Ioannina, Greece**
`bekos@uoi.gr, m.e.pavlidi@uoi.gr`

2 **University of Perugia, Italy**
`carla.binucci@unipg.it, emilio.digiacomo@unipg.it, walter.didimo@unipg.it, luca.grilli@unipg.it, alessandra.tappini@unipg.it`

3 **FernUniversität in Hagen, Germany**
`alexandra.weinberger@fernuni-hagen.de`

Abstract

A linear layout of a graph defines a total order of the vertices and partitions the edges into either *stacks* or *queues*, i.e., crossing-free and non-nested sets of edges along the order, respectively. In this work, we study defective linear layouts that allow forbidden patterns among edges of the same set. Our focus is on *k-defective stack layouts* and *k-defective queue layouts*, in which the *conflict graph* representing the forbidden patterns among the edges of each stack or queue has maximum degree k.

1 Introduction

Stack [5] and queue [8] layouts have been a central topic in topological graph theory. These layouts order the vertices and partition the edges into stacks (crossing-free sets of edges) or queues (non-nested sets of edges). Given a graph, the focus is on the *stack* or *queue number*, that is, the minimum number of stacks or queues, respectively, needed for a corresponding layout to exist. This work introduces *k-defective linear layouts*, a generalization that allows forbidden patterns (crossing edges in stacks, nested edges in queues) by means of a conflict graph that is required to have maximum vertex-degree k. Classical stack/queue layouts form special cases for $k = 0$. Our results include: (i) characterizations for k-defective queue/stack layouts; (ii) bounds on the edge density of k-defective queue/stack layouts; (iii) bounds or exact values for the k-defective queue/stack number of specific graph families, e.g., complete, outerplanar, and outer 1-planar graphs. Due to space constraints, some proofs are either outlined briefly or omitted.

2 Preliminaries

A vertex order $\prec$ of a graph G is a total order of its vertices. Two independent edges (u_1, v_1), (u_2, v_2) nest if $u_1 \prec u_2 \prec v_2 \prec v_1$, while they cross if $u_1 \prec u_2 \prec v_1 \prec v_2$. A stack (queue) is a set of pairwise non-crossing (non-nested) edges. An edge forms a k-defect if it crosses (nests or is being nested by) k others in the same stack (queue) in $\prec$. Accordingly, a set of edges, each of which forms a κ-defect with $\kappa \leq k$ is called k-defective stack (queue, respectively). For $h \geq 1$ and $k \geq 0$, a k-defective h-stack (queue) layout of a graph G consists of a vertex order and a partition of the edges into h defective stacks (queues), each avoiding

[*] Research started at the Bertinoro Workshop on Graph Drawing 2024.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.

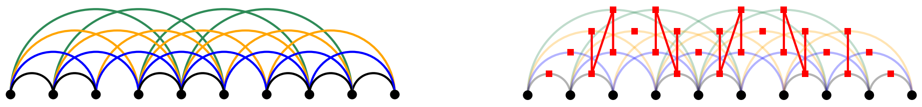


Figure 1 (Left) A 2-defective 1-queue layout of a 10-vertex graph. (Right) Its conflict graph.

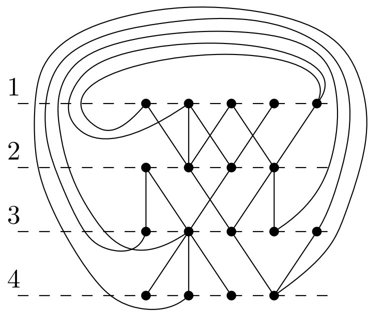


Figure 2 An arched level 1-planar layout of a graph.

$(k+1)$-defects. The k-defective stack (queue) number of G is the minimum h for such a layout to exist. Defective stack/queue layouts are related to defective colorings [2], in which adjacent vertices can share a color, as long as each monochromatic component has a specific structure as outlined in Property 1.

▶ Property 1. A graph has a k-defective h-stack/queue layout if and only if it admits a linear order $\prec$ whose conflict graph $C_\prec$ (having a node for each edge and an edge (u, v) exists in $C_\prec$ if and only if the edges corresponding to u and v form a 1-defect in $\prec$) has a defective h-coloring in which every monochromatic induced component has maximum degree at most k.

As a graph of maximum degree k is $(k+1)$-colorable, by Property 1, every graph with k-defective stack (queue) number h has stack (queue) number at most $(k+1)h$. Thus, if a graph has k-defective stack (queue) number 1, its stack (queue) number is at most $k+1$, but the converse does not hold. For example, K_n has stack (queue) number $\lceil \frac{n}{2} \rceil$[5] ($\lfloor \frac{n}{2} \rfloor$[8]), while any defective 1-stack/-queue layout of K_n has defectiveness greater than $\lceil \frac{n}{2} \rceil - 1$ ($\lfloor \frac{n}{2} \rfloor - 1$). Property 1 also has implications to layouts with a fixed linear order.

3 Queue layouts

Characterization. An *arched level k-planar layout* of a graph G is a drawing where vertices lie on $\ell \geq 1$ horizontal levels (i.e., lines), edges connect vertices on the same level (drawn as arcs) or consecutive levels (drawn as straight-line segments) and each edge has at most k crossings; see Fig. 2. By extending a known result [8], one can show that a graph has k-defective queue number 1 if and only if it admits an arched level k-planar layout.

Edge density. We next study the density of graphs admitting k-defective 1-queue layouts. Consider a linear order of the vertices of a graph G, denoted by $v_0 \prec \cdots \prec v_{n-1}$. A *right (left) alternating path* in $\prec$ is defined as a path $P = \langle v_{j_1}, v_{j_2}, \ldots, v_{j_l} \rangle$ where the indices alternate in direction: if i is odd (even), $j_{i+2} < j_i < j_{i+1}$; if i is even (odd), $j_{i+2} > j_i > j_{i+1}$, $0 \leq j_i, j_{i+1}, j_{i+2} \leq n-1$. The edges of a k-defective h-queue layout $\mathcal{L}$ of K_n can be partitioned into $n-1$ right (or left) alternating paths $P_1, P_2, \ldots, P_{n-1}$ such that each P_i has i

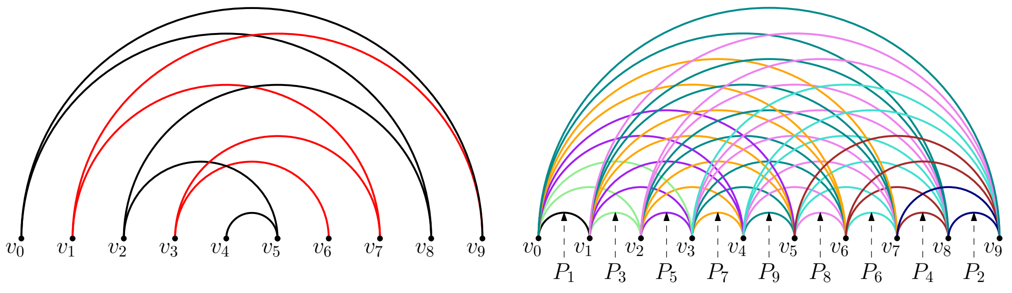


Figure 3 (Left) Two alternating paths; left one (red) and a right one (black). (Right) An alternating path partition of K_{10}.

edges ($1 \leq i \leq n-1$); see the right part of Figure 3. Such a partition is called *alternating path partition.*

▶ **Lemma 3.1.** *Let $\prec$ be the linear order of a k-defective h-queue layout $\mathcal{L}$ of a graph G, and let P be an alternating path of $\prec$. Every defective queue of $\mathcal{L}$ contains at most $k+2$ edges of P.*

Proof. If a defective queue contained a set E^* of more than $k+2$ edges of P, then the longest edge would nest all but one edge, making the defectiveness of $\mathcal{L}$ at least $k+1$. ◀

We next prove the upper bound on the density of graphs admitting k-defective 1-queue layouts.

▶ **Theorem 3.2.** *Every n-vertex graph that admits a k-defective 1-queue layout has at most $(k+2)n - \frac{(k+2)(k+3)}{2}$ edges, which is a tight bound for $k = 1$.*

Proof. Let $\mathcal{L}$ be a k-defective 1-queue layout of an n-vertex graph G, and let $\prec$ be its linear order. Graph G can have as many edges as K_n, which in turn admits an alternating path partition $P_1, P_2, \ldots, P_{n-1}$ as illustrated in the right part of Fig. 3. Since $\mathcal{L}$ has a single defective queue, by Theorem 3.1, graph G can have at most $k+2$ edges in each P_i, giving an upper bound of $(k+2)(n-1)$ edges. However, each P_i with $1 \leq i \leq k+1$ is missing $k+2-i$ edges to reach the upper bound of $k+2$. Thus, the maximum edge density is $(k+2)(n-1) - \sum_{j=1}^{k+1} j = (k+2)n - \frac{(k+2)(k+3)}{2}$. To prove that the bound is tight for $k=1$, for every $n \geq 3$ we construct a 1-defective 1-queue layout of a graph G with $3n-6$ edges. Let $v_0 \prec v_1 \prec \cdots \prec v_{n-1}$ be a linear order of the vertices of G. Add to G all the 1- 2- and 3-hop edges (black, blue, and orange, respectively, in Fig. 4). The constructed layout is a 1-defective 1-queue layout. Since the number of i-hop edges is $n-i$, with $i \in \{1,2,3\}$, graph G has in total $3n-6$ edges. ◀

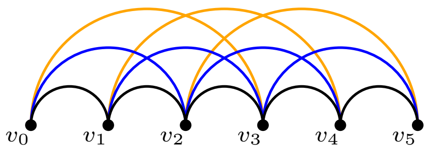


Figure 4 A 1-defective 1-queue layout with 6 vertices and 12 edges.

▶ **Theorem 3.3.** *For every integer $j \geq 1$, there exists an n-vertex graph with $n = 3j+1$ vertices and $\frac{10}{3}n - \frac{22}{3} = 10j - 4$ edges that admits a 2-defective 1-queue layout.*

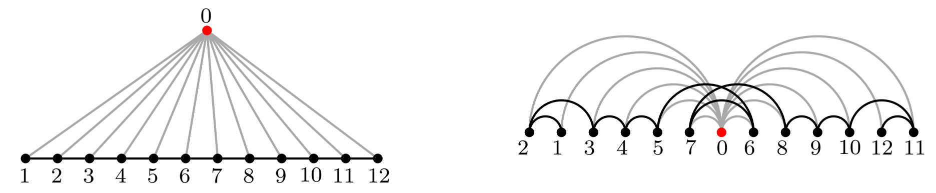


Figure 5 Illustration for the proof of Theorem 3.4

Sketch. Let $j \in \mathbb{Z}^+$ and $n = 3j+1$. A 2-defective 1-queue layout of a graph with n vertices and $\frac{10}{3}n - \frac{22}{3}$ edges consists of (see Fig. 1 for $n = 10$): (i) $\frac{2}{3}n + \frac{1}{3}$ 1-hop edges (black in Fig. 1); (ii) $n-2$ 2-hop edges (blue); (iii) $n-3$ 3-hop edges (orange); (iv) $\frac{2}{3}n - \frac{8}{3}$ 4-hop edges (green). Namely, in the complete left-to-right sequence of $(n-1)$ 1-hop edges and $(n-4)$ 4-hops, we delete the last edge from each triple of consecutive 1-hop edges, except for the last triple and the last edge from each triple of 4-hop edges. ◀

Defectiveness and Queue Number. Since outerplanar graphs have queue number 2 [9], it is natural to ask whether constant defectiveness allows every outerplanar graph to have k-defective queue number 1 for some constant k. We negatively answer this question below.

▶ **Lemma 3.4.** *For every integer $j \geq 3$, there is an outerplanar graph G with $n = 3j+1$ vertices such that every defective 1-queue layout of G has defectiveness at least $\frac{n-1}{3} - 2$.*

Sketch. Given $j \geq 3$, let G be an outerplanar graph with $n = 3j+1$ vertices consisting of a path of $n-1$ vertices (*backbone*) and a vertex (*apex*) connected to all backbone vertices (*fan edges*); see Fig. 5(left) for $j = 4$. Graph G has $m = 2n-3$ edges; $n-1$ fan edges and $n-2$ backbone edges. In any 1-defective queue layout $\mathcal{L}$ of G, let a be the position of the apex in $\prec$, with E_m, E_l, and E_r being subsets of the backbone edges nesting a, left of a, and right of a, respectively. $\mathcal{L}$ contains a k-defect such that $k \geq M = \max\{|E_l|-2, |E_m|-2, |E_r|-2\}$. M is minimized when two of E_l, E_m, and E_r have size $\frac{n-1}{3}$, and the other $\frac{n-1}{3} - 1$, yielding $M = \frac{n-1}{3} - 2$. A matching layout for $k = \frac{n-1}{3} - 2$ is given in Fig. 5(right), with $|E_l| = |E_r| = \frac{n-1}{3}$, $|E_m| = \frac{n-1}{3} - 1$, and degree-2 backbone vertices at both ends. ◀

Outer 1-planar graphs (that is, graphs admitting 1-planar drawings with all vertices being incident to the outer face) have (0-defective) queue number at most 42, as they are planar [4]. In the following, we show that their 1-defective queue number is 2, which implies that their (0-defective) queue number at most 3.

▶ **Theorem 3.5.** *Outer 1-planar graphs have 1-defective queue number 2.*

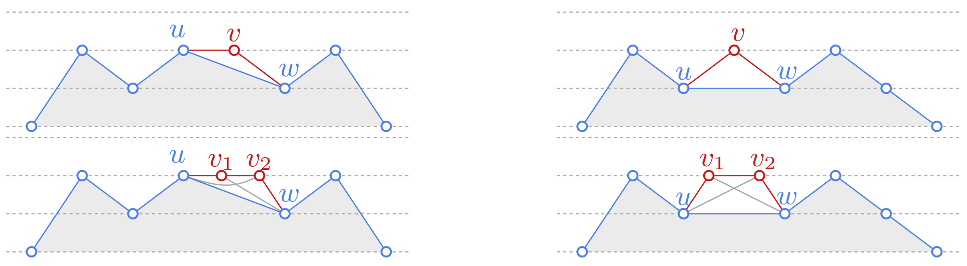


Figure 6 Illustration for the proof of Theorem 3.5.

Sketch. For the upper bound, we adapt a construction [3, 7] that yields weakly leveled planar drawings of outerplanar graphs, in which the edges have either span 0 (forming a forest of paths) or span 1 (connecting consecutive levels). More precisely, for an outer 1-planar graph G, we incrementally construct a planar straight-line drawing $\Gamma(G_p)$ of its planar skeleton G_p by introducing one or two vertices each time maintaining the following invariants (see Fig. 6): (i) the outer face is bounded by two strictly x-monotone paths (upper and lower envelopes), (ii) the end-vertices of the edges differ in y-coordinate by 0 (span-0 edges) or 1 (span-1 edges). Then, inserting the crossed edges of $G \setminus G_p$ into $\Gamma(G_p)$ yields a 1-planar drawing $\Gamma(G)$ that can be converted to a 1-defective 2-queue layout for G as follows: (i) $u \prec v$ if $y(u) > y(v)$ or $y(u) = y(v)$ and $x(u) < x(v)$ in $\Gamma(G)$, (ii) span-0 edges form the first (0-defective) queue, while span-1 its second (1-defective). ◀

▶ **Corollary 3.6.** *Outer 1-planar graphs have queue number at most* 3.

By using the fact that the queue number of planar 3-trees is at most 5 [1], Lemma 3.7, which forms an adaption of Lemma 8 in [6], and Lemma 3.8, we can prove that planar graphs have 1-defective queue number at most 33 (by setting $\ell = 3$ in Lemma 3.8). For the definitions of H-partitions and of BFS-layerings, we point the reader to [6].

▶ **Lemma 3.7.** *Let G be a graph admitting an H-partition of layered-width ℓ with respect to a BFS-layering $\mathcal{B}$. Then, the* 1*-defective queue number of G is upper-bounded by:*

$$(3 \cdot qn(H) + 1) \cdot \left\lceil \frac{2\ell}{3} \right\rceil + \left\lceil \frac{\ell - 1}{3} \right\rceil .$$

Sketch. The proof follows the same lines as the one in [6], where the bounds on the queue numbers of K_ℓ and $K_{\ell,\ell}$ are substituted by the 1-defective ones provided in the following. ◀

▶ **Lemma 3.8.** *The* 1*-defective queue number of $K_{n,n}$ in the separated setting (i.e., when all vertices of one part precedes those of the other) is at least $\lceil \frac{n}{2} \rceil$ and at most $\lceil \frac{2n}{3} \rceil$.*

Proof. Let $u_0, \ldots, u_{n-1}$ and $v_0, \ldots, v_{n-1}$ be the vertices of the two parts of $K_{n,n}$, ordered as $u_0 \prec \cdots \prec u_{n-1} \prec v_0 \prec \cdots \prec v_{n-1}$. The edges $(u_0, v_{n-1}), (u_1, v_{n-2}), \ldots, (u_{n-1}, v_0)$ pairwise nest, and since at most two can share a single 1-defective queue page, the lower bound follows. For the upper bound, assuming $n \bmod 3 = 0$, we prove that $K_{n,n}$ admits a 1-defective queue layout $\mathcal{L}_n$ with $\frac{2n}{3}$ 1-defective queues denoted by q_0 to $q_{\frac{2n}{3}-1}$.

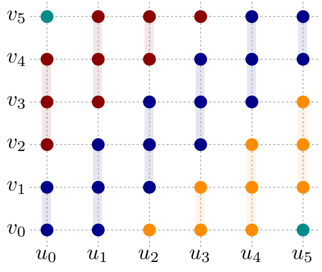


Figure 7 An assignment of the edges of $K_{6,6}$ to the four pages of a 1-defective queue layout.

Page q_0 contains only the two edges (u_0, v_{n-1}) and (u_{n-1}, v_0), while q_1 contains $3n - 2$ edges:

● (u_i, v_{i-1}) $1 \le i \le n-1$ ● (u_i, v_i) $0 \le i \le n-1$ ● (u_i, v_{i+1}) $0 \le i \le n-2$.

The remaining pages of $\mathcal{L}_n$ are defined as follows: for each $p \in [1, \frac{n}{3} - 1]$, $\mathcal{L}_n$ includes two symmetric pages, q_{2p} and q_{2p+1}, each with $3n - 9p$ edges. Then, q_{2p} contains the edges:

- ● (u_{i+3p+1}, v_i) $0 \le i \le n-3p-2$
- ● (u_{i+3p}, v_i) $0 \le i \le n-3p-1$
- ● (u_{i+3p-1}, v_i) $0 \le i \le n-3p$.

On the other hand, page q_{2p+1} contains the following edges:

- ● (u_i, v_{i+3p+1}) $0 \le i \le n-3p-2$
- ● (u_i, v_{i+3p}) $0 \le i \le n-3p-1$
- ● (u_i, v_{i+3p-1}) $0 \le i \le n-3p$.

Thus, $\mathcal{L}_n$ contains $3n + 2 \cdot \sum_{p=1}^{\frac{n}{3}-1}(3n-9p) = n^2$ edges. The proof relies on two points: no edge of $K_{n,n}$ is assigned to multiple pages, and no page of $\mathcal{L}_n$ has a k-defect for $k > 1$. ◀

In the separated setting, a k-defective queue layout can be converted into a k-defective stack one by reversing the vertex order of one bipartition. Hence, Theorem 3.8 directly implies:

▶ **Corollary 3.9.** *The* 1*-defective stack number of* $K_{n,n}$ *in the separated setting is at least* $\lceil \frac{n}{2} \rceil$ *and at most* $\lceil \frac{2n}{3} \rceil$.

In the following, we establish a lower (Theorem 3.10) and an upper bound (Theorem 3.11) on the k-defective queue number of K_n, which is tight for $k = 1$ (Corollary 3.12).

▶ **Theorem 3.10.** *The k-defective queue number of K_n is at least* $\left\lceil \frac{n-1}{k+2} \right\rceil$.

Proof. The edges of a k-defective queue layout $\mathcal{L}$ of K_n can be partitioned into alternating paths $P_1, \dots, P_{n-1}$. By Lemma 3.1, at most $k+2$ edges out of the $n-1$ edges of P_{n-1} can be in the same queue. Thus, at least $\left\lceil \frac{n-1}{k+2} \right\rceil$ queues are required in $\mathcal{L}$. ◀

The corresponding upper bound on the k-defective queue number of K_n that is given in the following theorem is more tedious to be obtained; its detailed proof is postponed for the conference or journal version of this note.

▶ **Theorem 3.11.** *The k-defective queue number of K_n is at most* $\lceil \frac{n-1}{l} \rceil$ *with* $l = \left\lfloor \frac{3+\sqrt{8k+1}}{2} \right\rfloor$.

▶ **Corollary 3.12.** *The* 1*-defective queue number of K_n is* $\lceil \frac{n-1}{3} \rceil$.

4 Stack Layouts

In this section, we turn our attention to k-defective h-stack layouts.

Characterization. It is not difficult to see that a graph admits a k-defective h-stack layout if and only if its edges can be partitioned into h defective stacks, each inducing an outer k-planar subgraph. We outline this observation in the following theorem.

▶ **Theorem 4.1.** *A graph has k-defective stack number* 1 *if and only if it is outer k-planar.*

Edge density. The following theorem provides an upper bound on the edge density of graphs admitting 1-defective h-stack layouts.

▶ **Theorem 4.2.** *An n-vertex graph that admits a* 1*-defective h-stack layout has at most* $(\frac{3}{2}h+1)n - 4h$ *edges.*

Proof. Let G be an n-vertex graph admitting a 1-defective h-stack layout $\mathcal{L}$. Let $v_0, \dots, v_{n-1}$ be the vertices of G in the order they appear in $\mathcal{L}$. An edge e *connects consecutive vertices* in $\mathcal{L}$ if $e = (v_i, v_{i+1})$ for some $i \in \{0, \dots, n-1\}$, with indices taken mod n. By Theorem 4.1, each defective stack of $\mathcal{L}$ is an outer 1-planar graph with at most $\frac{3}{2}n - 4$ edges, excluding the at most n edges connecting consecutive vertices. Since $\mathcal{L}$ has h stacks, G has at most $\frac{3}{2}hn - 4h$ edges. With the consecutive edges, G has at most $\left(\frac{3}{2}h + 1\right)n - 4h$ edges. ◀

Defectiveness and stack number. We conclude this section by presenting upper and lower bounds on the k-defective stack number of K_n. In particular, the following theorems can be proved using similar arguments as the corresponding ones for defective queue layouts.

▶ **Theorem 4.3.** *The* 1*-defective stack-number of* K_n *is either* $\lfloor \frac{n}{3} \rfloor - 1$ *or* $\lceil \frac{n}{3} \rceil$.

▶ **Theorem 4.4.** *The* k*-defective stack number of* K_n *is at most* $\lceil \frac{n}{l+2} \rceil$*, where* $l{=}\lfloor \frac{-1+\sqrt{8k+1}}{2} \rfloor$.

5 Open Problems

Our work raises several new open problems, which we list below.

- Even though we focused on stack and queue layouts, the study can naturally extend to other types of linear layouts (e.g., deques or riques) or even to mixed settings.
- In relation to the type of defects, we considered those obtained by imposing restrictions on the degree of the conflict graph. It would be interesting to study other types of defects, e.g., bounded diameter for the conflict graph.
- More specific research questions raised directly from our results include (i) proving hardness for recognizing the graphs that admit k-defective 1-queue layouts when $k \leq 3$, (ii) improving our bounds on the k-defective stack and queue number of K_n when $k > 1$, and (iii) extending the study to other classes of graphs.

References

1 Jawaherul Md. Alam, Michael A. Bekos, Martin Gronemann, Michael Kaufmann, and Sergey Pupyrev. Queue layouts of planar 3-trees. *Algorithmica*, 82(9):2564–2585, 2020. `doi:10.1007/s00453-020-00697-4`.

2 Patrizio Angelini, Michael A. Bekos, Felice De Luca, Walter Didimo, Michael Kaufmann, Stephen G. Kobourov, Fabrizio Montecchiani, Chrysanthi N. Raftopoulou, Vincenzo Roselli, and Antonios Symvonis. Vertex-coloring with defects. *J. Graph Algorithms Appl.*, 21(3):313–340, 2017. `doi:10.7155/JGAA.00418`.

3 Michael J. Bannister, William E. Devanny, Vida Dujmovic, David Eppstein, and David R. Wood. Track layouts, layered path decompositions, and leveled planarity. *Algorithmica*, 81(4):1561–1583, 2019. `doi:10.1007/s00453-018-0487-5`.

4 Michael A. Bekos, Martin Gronemann, and Chrysanthi N. Raftopoulou. An improved upper bound on the queue number of planar graphs. *Algorithmica*, 85(2):544–562, 2023. `doi:10.1007/s00453-022-01037-4`.

5 Frank Bernhart and Paul C. Kainen. The book thickness of a graph. *J. Comb. Theory, Ser. B*, 27(3):320–331, 1979. `doi:10.1016/0095-8956(79)90021-2`.

6 Vida Dujmovic, Gwenaël Joret, Piotr Micek, Pat Morin, Torsten Ueckerdt, and David R. Wood. Planar graphs have bounded queue-number. *J. ACM*, 67(4):22:1–22:38, 2020. `doi:10.1145/3385731`.

7 Stefan Felsner, Giuseppe Liotta, and Stephen K. Wismath. Straight-line drawings on restricted integer grids in two and three dimensions. *J. Graph Algorithms Appl.*, 7(4):363–398, 2003. URL: `https://doi.org/10.7155/jgaa.00075`, `doi:10.7155/JGAA.00075`.

8 Lenwood S. Heath and Arnold L. Rosenberg. Laying out graphs using queues. *SIAM J. Comput.*, 21(5):927–958, 1992. `doi:10.1137/0221055`.

9 S. Rengarajan and C. E. Veni Madhavan. Stack and queue number of 2-trees. In Ding-Zhu Du and Ming Li, editors, *Computing and Combinatorics, First Annual International Conference, COCOON '95, Xi'an, China, August 24-26, 1995, Proceedings*, volume 959

of *Lecture Notes in Computer Science*, pages 203–212. Springer, 1995. URL: `https://doi.org/10.1007/BFb0030834`, `doi:10.1007/BFB0030834`.

Faces of maximal plane graphs without short cycles[*]

Maria Axenovich[1], Leon Kießle[1], and Arsenii Sagdeev[1]

1 **Karlsruhe Institute of Technology, Karlsruhe, Germany**
maria.aksenovich@kit.edu, leon.kiessle@student.kit.edu,
sagdeevarsenii@gmail.com

Abstract

For a positive integer g, we study a family of plane graphs G without cycles of length less than g that are maximal in a sense that adding any new edge to G either makes it non-plane or creates a cycle of length less than g. We show that the largest face length $\mathrm{f}_{\max}(g)$ of a 2-connected graph from this family satisfies $3g - 12 \le \mathrm{f}_{\max}(g) \le 2(g-2)^2 + 1$.

Related Version arXiv:2410.13481

1 Introduction

Turán-type problems play a substantial role in combinatorics since their introduction by Mantel [15] and Turán [20] in the first half of the 20th century. Perhaps the most extensively studied question of this type is the following. For a given family $\mathcal{F}$ of graphs, what is the largest possible number of edges $\mathrm{ex}(n, \mathcal{F})$ in an n-vertex $\mathcal{F}$-*free* graph, that is, a graph that does not contain any $F \in \mathcal{F}$ as a subgraph? Let $\mathcal{C}_{<g} = \{C_3, \dots, C_{g-1}\}$ be a family of cycles of length less than g. It is known, for a fixed g, that $\mathrm{ex}(n, \mathcal{C}_{<g}) = O(n^{1+1/\lfloor (g-1)/2 \rfloor})$ and the bound is known to be asymptotically tight for some values of g, see [2, 8, 10].

Problems of this kind have a rich history of study and numerous variations, see the surveys [10, 19, 21]. Another variation was suggested by Dowden [7], who asked for the largest possible number of edges $\mathrm{ex}_{\mathcal{P}}(n, \mathcal{F})$ in an n-vertex *plane* $\mathcal{F}$-free graph. A direct application of Euler's formula yields that $\mathrm{ex}_{\mathcal{P}}(n, \mathcal{C}_{<g}) \le \frac{g}{g-2}(n-2)$ for all $n \ge g \ge 3$ which is essentially tight. For more partial results on Dowden's problem, we refer the reader to [5, 11, 12, 14, 17, 18] and the references therein.

In this note, we consider another natural parameter of a plane graph G, the largest face length, that we denote by $\mathrm{f}_{\max}(G)$. In other words, let $\mathrm{f}_{\max}(G)$ be the length of the longest cycle bounding a face of G. We say that a plane $\mathcal{C}_{<g}$-free graph G is *maximal plane* $\mathcal{C}_{<g}$-*free graph* if adding any new edge to G with both endpoints in G either makes it non-plane or creates a cycle of length less than g. The case when G is a star shows that $\mathrm{f}_{\max}(G)$ can be arbitrary large in terms of g even if our graph is maximal plane $\mathcal{C}_{<g}$-free. To avoid this rather degenerate situation, we consider only 2-connected graphs. More formally, let

$$\mathrm{f}_{\max}(g) = \max\{\mathrm{f}_{\max}(G) : \ G \text{ is a 2-connected maximal plane } \mathcal{C}_{<g}\text{-free graph}\}.$$

By taking $G = C_{2g-3}$, we immediately see that $\mathrm{f}_{\max}(g) \ge 2g - 3$. It is not hard to show that this inequality is tight for $g = 3, 4, 5$, while the first author, Ueckerdt, and Weiner [3, Lemma 5] proved this for $g = 6$. Our main result provides general upper and lower bounds on $\mathrm{f}_{\max}(g)$. In particular, it implies that $\mathrm{f}_{\max}(g)$ is finite and strictly larger than $2g - 3$ for all $g \ge 7$.

* Research was supported in part by the DFG grant FKZ AX 93/2-1.

▶ **Theorem 1.1.** *If* $3 \leq g \leq 6$*, then* $\mathrm{f}_{\max}(g) = 2g - 3$*. If* $7 \leq g \leq 9$*, then* $\mathrm{f}_{\max}(g) \geq 3g - 9$*. Moreover, for any integer* $g \geq 7$*, we have*

$$3g - 12 \leq \mathrm{f}_{\max}(g) \leq 2(g-2)^2 + 1.$$

We prove the general lower and upper bounds in Section 3. Some additional definitions are given in Section 2. Finally, we give concluding remarks in Section 4.

Let us note that the relations between face lengths in plane graphs and their other parameters including radius or diameter were also considered, see Ali, Dankelmann, Mukwembi [1] and Du Preez [16], respectively. See also a paper by Fernández, Sieger, and Tait [9] on planar subgraphs of given girth in planar graphs.

2 Definitions and basic observations

When clear from the context, we shall identify a *plane* graph with the corresponding *planar* one. For all standard graph theoretic notions, we refer the reader to a book by Diestel [6]. We call a path S in a graph G an *ear* of a cycle C in G if the endpoints of S are vertices of C and no other vertex or edge of S belongs to C. We say that an ear S *splits* C into paths C' and C'' if $C = C' \cup C''$ and the endpoints of C' and C'' are those of S. Moreover, for every two vertices x and y of G, let us denote the length of a shortest x, y*-path*, i.e., a path between x and y, by $\mathrm{dist}_G(x, y)$. When the graph under consideration is clear from the context, we simply write $\mathrm{dist}(x, y)$. We denote the set of integers $\{1, \dots, n\}$ by $[n]$.

We say that a graph is a *subdivided wheel* if it is a union of a cycle C, called the *outer cycle of the wheel*, and a tree T with exactly one vertex c, called the *center of the wheel*, of degree more than 2 such that a vertex of T belongs to C if and only if it is a leaf of T. We say that a path in T connecting c to a leaf of T is a *spoke of the wheel* and a path in C connecting two consecutive leaves of T is a *segment of the wheel.* Note that a maximal plane $\mathcal{C}_{<g}$-free graph $W(g)$ in Figure 1 (a) is a subdivided wheel.

We shall repeatedly use the following observation.

▶ **Lemma 2.1.** *Let* $g \geq 4$*,* G *be a 2-connected maximal plane* $\mathcal{C}_{<g}$*-free graph, and* C *be its facial cycle. If* x *and* y *are two non-consecutive vertices of* C*, then* $2 \leq \mathrm{dist}(x, y) \leq g - 2$.

Proof. Let x and y be two non-consecutive vertices on C.

If x and y are not adjacent, adding the edge xy to G doesn't break planarity, since x and y belong to the same face of G. Now the maximality of G implies that this new edge xy belongs to a cycle of length at most $g - 1$, and thus $\mathrm{dist}_G(x, y) \leq g - 2$, as desired.

Assume now that x and y are adjacent. This edge xy splits C into two paths C' and C'', each of length at least $g - 1$, since otherwise G contains a cycle of length less than g. Pick two vertices, x' on C' and y' on C'', such that $\mathrm{dist}_C(x, x') = \lfloor g/2 \rfloor = \mathrm{dist}_C(y, y')$. Note that $\mathrm{dist}(x, x') = \lfloor g/2 \rfloor$ otherwise the union of a shortest x, x'-path in G and the x, x'-path in C contains a cycle of length at most $g - 1$. Let P be a shortest x, x'-path. Assume that $\mathrm{dist}(x', y) < \lceil g/2 \rceil - 1$, i.e., that there is an x', y-path P' of length at most $\lceil g/2 \rceil - 2$. Then P' is shorter than P, and thus $P \cup P' \cup xy$ contains a cycle of length less than g. This is a contradiction implying that $\mathrm{dist}(x', y) \geq \lceil g/2 \rceil - 1$. Similarly, $\mathrm{dist}(y, y') = \lfloor g/2 \rfloor$ and $\mathrm{dist}(x, y') \geq \lceil g/2 \rceil - 1$. Since each x', y'-path contains either x or y by planarity, we conclude that $\mathrm{dist}(x', y') \geq \lfloor g/2 \rfloor + \lceil g/2 \rceil - 1 = g - 1$, which contradicts the first part of the lemma and thus completes the proof. ◀

3 Proof of Theorem 1.1 for $g \geq 7$

Let $g \geq 7$. For the general lower bound, consider the graph $W(g)$, see Figure 1 (a), that is a subdivided wheel with three spokes of length two each and each segment of length $g-4$. Observe that any two non-adjacent vertices of $W(g)$ belong to at least one of the three cycles of length $2g-4$, and thus adding an edge between them creates a cycle of length less than g. Hence, $W(g)$ is a 2-connected maximal plane $\mathcal{C}_{<g}$-free graph. Therefore, we have $\mathrm{f}_{\max}(g) \geq \mathrm{f}_{\max}(W(g)) = 3g-12$, as claimed.

If $g = 7, 8$, or 9, consider a different construction $W'(g)$, that is an edge disjoint union of C_9 and C_{3g-9} that share three vertices equidistant on each of the cycles, see Figure 1 (b). As earlier, note that any two non-adjacent vertices of $W'(g)$ belong either to at least one of the three cycles of length $2g-3$ or to at least one of the three cycles of length g. Hence, $W'(g)$ is a 2-connected maximal plane $\mathcal{C}_{<g}$-free graph, and so $\mathrm{f}_{\max}(g) \geq \mathrm{f}_{\max}(W'(g)) = 3g-9$ for $g = 7, 8, 9$, as claimed.

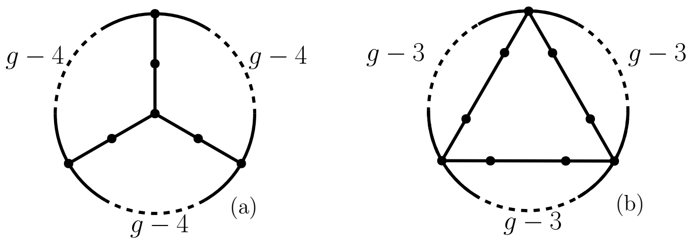


Figure 1 2-connected maximal plane $\mathcal{C}_{<g}$-free graphs $W(g)$ and $W'(g)$.

Now we shall give a general upper bound. Let G be a 2-connected maximal plane $\mathcal{C}_{<g}$-free graph, and C be its largest facial cycle. The main idea of the proof is as follows. We argue that if C is sufficiently long, then there is an ear S that splits it into two sufficiently long paths. Then we show that there are two vertices, one on each path, each at a distance more than $g/2-1$ to S. As a result, these two vertices are at a distance more than $g-2$ in G, which contradicts Lemma 2.1.

Let $S = v_1 \dots v_{k+1}$ be an ear of C that splits it into $C' = v_1 u_1 \dots u_{m-1} v_{k+1}$ and $C'' = v_1 u'_1 \dots u'_{m'-1} v_{k+1}$. Our goal is to bound $\mathrm{dist}_C(v_1, v_{k+1}) = \min(m, m')$ in terms of g and k, that we shall state in Corollary 3.5. To do so, we order the vertices of S according to their indices, i.e., we say that $v_i < v_{i'}$ if $i < i'$. Let

$$d(j) = \mathrm{dist}(u_j, S) = \min_{i \in [k+1]} \mathrm{dist}(u_j, v_i) \qquad \text{and}$$

$$v(j) = v_i, \quad \text{where } i = \min\{i' : d(j) = \mathrm{dist}(u_j, v_{i'})\},$$

i.e., $v(j)$ is the smallest vertex from S such that $\mathrm{dist}(u_j, S) = \mathrm{dist}(u_j, v(j))$. Let $P(j)$ be some shortest $u_j, v(j)$-path. Note that each $P(j)$ is in the region bounded by C' and S, see Figure 2. For a vertex v of S, let

$$J(v) = \{j \in [m-1] : v(j) = v\}.$$

▶ **Lemma 3.1.** *If $d(j) \leq g/2-1$ for each $j \in [m-1]$, then $m \leq k(g-3)+2$.*

Proof. Our argument consists of several steps.

▶ Claim 3.2. For any $j \in [m-2]$, $v(j) \leq v(j+1)$. In particular, for any vertex v of S, $J(v)$ is an interval of consecutive integers, that also could be empty.

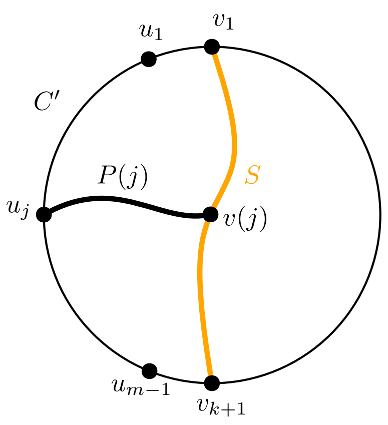


Figure 2 Setting for the proof of the general upper bound on $\mathrm{f}_{\max}(g)$.

Indeed, assume that $v(j+1) < v(j)$ for some j. Since the edges of $P(j)$ and $P(j+1)$ are in the same region bounded by C' and S, by planarity, $P(j)$ and $P(j+1)$ share a vertex, say x. Let $P'(j)$ be the subpath of $P(j)$ from x to $v(j)$, define $P'(j+1)$ similarly.

Note that if $P'(j)$ is shorter than $P'(j+1)$, then $P(j+1)$ is not a shortest path from u_{j+1} to S since 'exchanging' $P'(j+1)$ to $P'(j)$ in $P(j+1)$ yields a shorter walk.

Otherwise, if $P'(j)$ is not shorter than $P'(j+1)$, then 'exchanging' $P'(j)$ to $P'(j+1)$ in $P(j)$ would yield a walk from u_j to $v(j+1)$ that is not longer than $P(j)$ which contradicts the definition of $v(j)$ since $v(j+1) < v(j)$. This proves the claim.

▶ Claim 3.3. For each vertex v of S, the function $d(\cdot)$ is strictly unimodal on $J(v)$. Moreover, $d(\cdot)$ is strictly monotone on $J(v_1)$ and on $J(v_{k+1})$.

Fix a vertex v in S. If $j, j+1 \in J(v)$, then $P(j) \cup P(j+1) \cup u_j u_{j+1}$ is a closed walk of length less than g. Hence, this walk is trivial, i.e., either $P(j) = u_j u_{j+1} \cup P(j+1)$ (in which case $d(j) = d(j+1)+1$) or $P(j+1) = u_{j+1}u_j \cup P(j)$ (in which case $d(j+1) = d(j)+1$). Assume that there is a local maximum of $d(\cdot)$ at $j \in J(v)$, i.e., that $d(j-1) = d(j) - 1 = d(j+1)$ for some $j-1, j, j+1 \in J(v)$. Then $P(j) = u_j u_{j-1} \cup P(j-1) = u_j u_{j+1} \cup P(j+1)$. This is a contradiction, since $P(j)$ has an endpoint u_j and two edges incident to it. Therefore, the function $d(\cdot)$ has a unique minimum on $J(v)$, it first strictly decreases and then strictly increases. Moreover, note that $\mathrm{dist}(u_1, v_1) = \mathrm{dist}(u_{m-1}, v_{k+1}) = 1$, and thus the corresponding minimums of $d(\cdot)$ on $J(v_1)$ and on $J(v_{k+1})$ are their endpoints $j = 1$ and $j = m-1$, respectively. For the illustration, see Figure 3. This proves the claim.

▶ Claim 3.4. For each vertex v of S, $|J(v)| \le g-3$. Moreover $|J(v_1)|, |J(v_{k+1})| \le g/2 - 1$.

The second part is immediate from the fact that $d(\cdot)$ can have values only in $[g/2-1]$ and strictly monotone on $J(v_1)$ and on $J(v_{k+1})$. For the first part, note that the strict unimodality of $d(\cdot)$ on $J(v)$ implies that $|J(v)| \le 2(g/2-1) - 1 = g-3$ for each $v \in V(S)$. This proves the claim.

Since each $j \in [m-1]$ belongs to $J(v)$, for some v and the sets $J(v)$ are pairwise disjoint, we have that $m - 1 = \sum_{i=1}^{k+1} |J(v_i)| \le 2(g/2-1) + (k-1)(g-3) = k(g-3)+1$, which completes the proof of Lemma 3.1. ◀

▶ **Corollary 3.5.** *Each ear $S = v_1 \dots v_{k+1}$ of C satisfies* $\mathrm{dist}_C(v_1, v_{k+1}) \le k(g-3)+2$.

Proof. Indeed, if not, then some ear S splits C into two paths C' and C'', each of length more than $k(g-3)+2$. By Lemma 3.1, there are vertices x on C' and y on C'' such that

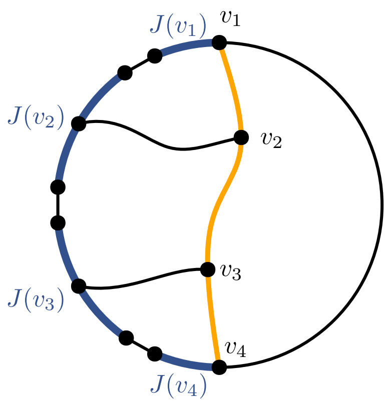


Figure 3 Special case in which the ear S (yellow) consists of 4 vertices; blue segments correspond to the sets $\{u_j : j \in J(v_i)\}$, $i \in [4]$; middle vertices on two blue segments correspond to the unique minimums of $d(\cdot)$ on these segments.

$\mathrm{dist}(x, S) > g/2 - 1$ and $\mathrm{dist}(y, S) > g/2 - 1$. Since every x, y-path goes through S by planarity, we conclude that $\mathrm{dist}(x, y) > g - 2$, which contradicts Lemma 2.1. ◀

Recall that C is a facial cycle of length $\mathrm{f}_{\max}(g)$, and take two "antipodal" vertices w and w' on it. Consider a w, w'-path P of length at most $g-2$, which exists by Lemma 2.1, and label the subsequence of its vertices that belong to C by $w = w_1, \dots, w_{t+1} = w'$. For $s \in [t]$, let $k_s - 1$ be the number of vertices of P strictly between w_s and w_{s+1}. Observe that $\sum_{s=1}^{t} k_s \le g - 2$ since this sum equals the length of P. Besides, if $k_s = 1$, then Lemma 2.1 implies that $\mathrm{dist}_C(w_s, w_{s+1}) = 1$. Moreover, note that if $k_s \ge 2$, then the w_s, w_{s+1}-subpath of P forms an ear of C, and thus $\mathrm{dist}_C(w_s, w_{s+1}) \le k_s(g-3) + 2$ by Corollary 3.5, see Figure 4. Finally, the triangle inequality implies that

$$\begin{aligned}
\mathrm{dist}_C(w, w') \le \sum_{s=1}^{t} \mathrm{dist}_C(w_s, w_{s+1}) &\le \sum_{s\colon k_s \ge 2} \big(k_s(g-3) + 2\big) + \sum_{s\colon k_s = 1} 1 \\
&= (g-3) \sum_{s\colon k_s \ge 2} k_s + \Big(\sum_{s\colon k_s \ge 2} 2 + \sum_{s\colon k_s = 1} 1 \Big) \\
&\le (g-3) \sum_{s=1}^{t} k_s + \sum_{s=1}^{t} k_s = (g-2) \sum_{s=1}^{t} k_s \le (g-2)^2.
\end{aligned}$$

Recall that w and w' are antipodal vertices of the largest face of G, and thus

$$\mathrm{f}_{\max}(G) \le 2 \cdot \mathrm{dist}_C(w, w') + 1 \le 2(g-2)^2 + 1,$$

which completes the proof of Theorem 1.1 for $g \ge 7$.

4 Concluding remarks

In this note we showed that the largest face length $\mathrm{f}_{\max}(g)$ of a 2-connected plane graph of girth at least g that is edge-maximal with respect to these two properties satisfies $\Omega(g) = \mathrm{f}_{\max}(g) = O(g^2)$. We would like to pose the following question.

▶ Question 4.1. Is it true that $\mathrm{f}_{\max}(g) = \Theta(g)$?

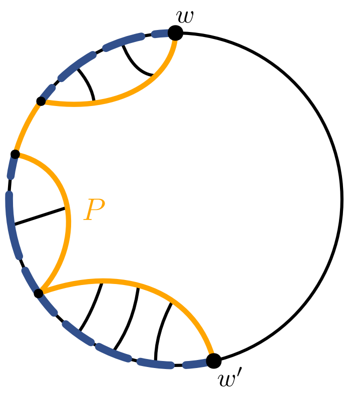


Figure 4 A path P between two antipodal vertices w and w' contains several ears of C; we apply Corollary 3.5 to bound the cyclic distance between the endpoints of each ear.

While determining $\mathrm{f}_{\max}(g)$ remains most interesting question of this note, it was not originally obvious that $\mathrm{f}_{\max}(g)$ is bounded by any function of g. The shortest argument we know takes about a page and is based on multicolor Ramsey's theorem applied to the auxiliary complete graphs, where we color the edge xy according to $\mathrm{dist}_G(x, y)$ for all vertices x, y of G. It might be interesting to find a shorter argument.

Another possible direction of research would be to relax the condition that G must be 2-connected in the definition of $\mathrm{f}_{\max}(g)$. As we mentioned in the introduction, the case when G is a (subdivided) star shows that $\mathrm{f}_{\max}(G)$ can be arbitrary large in terms of g even if our graph is maximal plane $\mathcal{C}_{<g}$-free. Are there examples of this sort that do not have leaves?

Acknowledgments. We thank the anonymous reviewers for their valuable suggestions.

References

1 P. Ali, P. Dankelmann, S. Mukwembi, The radius of k-connected planar graphs with bounded faces, Discrete Mathematics 312 (2012) 3636–3642.

2 N. Alon, S. Hoory, N. Linial, The Moore bound for irregular graphs, Graphs and Combinatorics 18 (2002) 53–57.

3 M. Axenovich, T. Ueckerdt, P. Weiner, Splitting planar graphs of girth 6 into two linear forests with short paths, Journal of Graph Theory 85(3) (2017) 601–618.

4 L. Caccetta, K. Vijayan, Maximal cycles in graphs, Discrete Mathematics 98 (1991) 1–7.

5 D. W. Cranston, B. Lidický, X. Liu, A. Shantanam, Planar Turán numbers of cycles: a counterexample, Electronic Journal of Combinatorics 29(3) (2022).

6 R. Diestel, Graph Theory, Graduate Texts in Mathematics, vol 173. Springer, Berlin, Heidelberg (2017).

7 C. Dowden, Extremal C_4-free, C_5-free planar graphs, Journal of Graph Theory 83(3) (2016) 213–230.

8 R. Dutton, R. Brigham, Edges in graphs with large girth, Graphs and Combinatorics 7(4) (1991) 315–321.

9 M. Fernández, N. Sieger, M. Tait, Maximal planar subgraphs of fixed girth in random graphs, Electronic Journal of Combinatorics 25(2) (2018).

10 Z. Füredi, M. Simonovits, The history of degenerate (bipartite) extremal graph problems, Erdős centennial, Bolyai Soc. Math. Stud., vol. 25, Bolyai Math. Soc., Budapest (2013) 169–264.

11 D. Ghosh, E. Győri, R. R. Martin, A. Paulos, C. Xiao, Planar Turán number of the 6-cycle, SIAM Journal on Discrete Mathematics 36(3) (2022) 2028–2050.

12 E. Győri, A. Li, R. Zhou, The planar Turán number of the seven-cycle, arXiv preprint arXiv:2307.06909 (2023).

13 L. Kießle, Bounding the maximum face length of girth-planar maximal graphs, Bachelor thesis, Karlsruhe Institute of Technology, 2024.

14 Y. Lan, Y. Shi, Z.-X. Song, Extremal H-free planar graphs, Electronic Journal of Combinatorics 26(2) (2019).

15 W. Mantel, Problem 28, in: Wiskundige Opgaven 10. Band 10 (1907) 60–61.

16 B. Du Preez, Plane graphs with large faces and small diameter, Australasian Journal of Combinatorics, 80(3) (2021) 401–418.

17 R. Shi, Z. Walsh, X. Yu, Dense circuit graphs and the planar Turán number of a cycle, Journal of Graph Theory, first online (2024).

18 R. Shi, Z. Walsh, X. Yu, Planar Turán number of the 7-cycle, arXiv preprint arXiv:2306.13594 (2023).

19 A. Sidorenko, What we know and what we do not know about Turán numbers, Graphs and Combinatorics 11(2) (1995) 179–199.

20 P. Turán, Eine Extremalaufgabe aus der Graphentheorie, Mat. Fiz. Lapok 48 (1941) 436–452.

21 J. Verstraete, Extremal problems for cycles in graphs, Recent trends in combinatorics. Cham: Springer International Publishing (2016) 83–116.

An Order for Higher-Dimensional Simplex Sweeps

Tim Ophelders[1] **and Anna Schenfisch**[2]

1 **Department of Information and Computing Sciences, Utrecht University, the Netherlands and Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands**
`t.a.e.ophelders@uu.nl`

2 **Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands**
`a.k.schenfisch@tue.nl`

Abstract

Standard sweep algorithms require an order of discrete points in Euclidean space, and rely on the property that, at a given point, all points in the halfspace below come earlier in this order. We generalize this notion by defining an order on i-simplices, maintaining the property that, at a given i-simplex σ, all $(i+1)$-dimensional cofaces of σ in the halfspace below σ have an i-dimensional face that appeared earlier in the order ("below" with respect to a particular direction perpendicular to σ). Such a sweeping order is computable even for degenerate simplicial complexes. In the full version of our work, we utilize this order in our motivating problem, namely, extending a sweep-based graph reconstruction algorithm to simplicial complex reconstruction.

Related Version `https://arxiv.org/pdf/2501.01901`

1 Introduction

Suppose we want to efficiently reconstruct an unknown simplicial complex K in Euclidean space by querying local structure. Specifically, for a query simplex σ of K and a query direction, the local structure we use is the *indegree* of σ in that direction: the number of cofacets of σ contained in some halfspace "below" σ. We assume that the vertex set of K is known. For the special case where K is a graph, [2] addresses this problem via a sweep algorithm through the vertex set, using indegree to identify edges adjacent to and above each vertex in the sweeping order. The algorithm relies on a natural but crucial property of the sweeping order: at a given vertex v, all edges adjacent to v contained in the halfspace below v have another endpoint that appeared earlier in the order.

The methods of [2] do not immediately extend to higher-dimensional simplex reconstruction. In particular, it is not possible to sweep through a set of i-simplices in a fixed direction and maintain that all $(i+1)$-cofacets of a given simplex σ that come below σ are known. We circumvent this by defining a *sweeping order* on a set of i-simplices, that, instead of using a fixed direction, additionally pairs each i-simplex σ with a direction perpendicular to σ. The full version of our work contains further details, as well as an application of such an order, namely, extending the edge reconstruction algorithm of [2] to simplicial complex reconstruction.

2 Preliminaries

In this section, we define our most extensively used terms, and refer the reader to [1, 3] for further information. We start with simplices and simplicial complexes.

▶ **Definition 2.1** (Simplex)**.** An *abstract i-simplex* σ is a set of $i+1$ elements, called vertices. The dimension of σ is i, denoted $\dim(\sigma)$. If σ and τ are abstract simplices and $\sigma \subseteq \tau$, we call σ a *face* of τ and τ a *coface* of σ. If $\dim(\sigma) = \dim(\tau) - 1$, we call σ a *facet* of τ and τ a *cofacet* of σ. An *i-simplex in $\mathbb{R}^d$* is an abstract i-simplex where each vertex maps to a distinct point in $\mathbb{R}^d$. We geometrically interpret a simplex as the convex hull of these points.

▶ **Definition 2.2** (Simplicial Complex)**.** An abstract simplicial complex K is a set of abstract simplices, such that if $\sigma \in K$ and $\rho \subseteq \sigma$, then $\rho \in K$. A *simplicial complex in $\mathbb{R}^d$* consists of an abstract simplicial complex, where each vertex is mapped injectively to a point in $\mathbb{R}^d$. Geometrically, we think of the simplicial complex as the union of convex hulls of its simplices.

Note that this allows degeneracies. We always view simplicial complexes geometrically, so we may be less precise with terminology going forward. Let K be a simplicial complex in $\mathbb{R}^d$. We always take $d \geq 2$. Let $\dim(K)$ denote the maximum dimension over simplices in K. Denote the i-skeleton of K by K_i and the number of i-simplices by n_i. For an i-simplex σ, let $\operatorname{cof}(\sigma) \subseteq K_{i+1}$ be the set of cofacets of σ and $\operatorname{aff}(\sigma)$ be the *affine hull* of σ.

The set of all unit vectors in $\mathbb{R}^d$ is parameterized by the unit $(d-1)$-sphere, $\mathbb{S}^{d-1}$, and a unit vector is called a *direction*. Denote by $\perp_\sigma \subseteq \mathbb{S}^{d-1}$ the set of directions perpendicular to $\operatorname{aff}(\sigma)$ and note if $\dim(\operatorname{aff}(\sigma)) = i' \leq d-1$, then $\perp_\sigma$ is a $(d-i'-1)$-sphere. With respect to some $s \in \perp_\sigma$, all points in σ have the same height, which we refer to as the *s-coordinate* of σ. We may abuse dot product notation and write $s \cdot \sigma$ to denote this s-coordinate. For $s \in \perp_\sigma$, the set of *down-cofacets* of σ, denoted $\operatorname{cof}_s^{<}(\sigma) \subseteq \operatorname{cof}(\sigma)$, is the set of cofacets $\sigma \cup \{v\}$ for which $s \cdot v < s \cdot \sigma$, i.e., v lies in the open halfspace below σ with respect to direction s.

Conceptually, our algorithms involve ordering points by rotating a hyperplane around some central space. In the following definition, we ensure that we make a consistent choice of normal direction to associate with each point we encounter.

▶ **Definition 2.3** (γ-Normal of p)**.** Consider a unit circle $S \subset \mathbb{R}^d$, angularly parameterized by $\gamma\colon [0, 2\pi) \to S$ and centered at some point $c \in \mathbb{R}^d$. For a point $p \in \mathbb{R}^d$, let $q \in \mathbb{R}^d$ be the orthogonal projection of p onto the plane containing γ. If $q \neq c$, let α be the unique angle such that the ray from c through $\gamma(\alpha - \pi/2 \bmod 2\pi)$ passes through q. We call $\gamma(\alpha)$ the *γ-normal of p relative to c*. If $q = c$, we define the γ-normal of p to be $\gamma(0)$.

Now consider a simplex $\sigma \subset \mathbb{R}^d$ with $\dim(\operatorname{aff}(\sigma)) < d-1$, let γ be an angularly parameterized circle of directions all perpendicular to σ, and let p be a point in $\mathbb{R}^d$. Then, the γ-normal of p relative to any point c in $\operatorname{aff}(\sigma)$ is the same, so we unambiguously define the *γ-normal of p relative to σ* to be the γ-normal of p relative to any such point c. See Figure 1 (a) and (b).

Finally, consider a simplex $\sigma \subset \mathbb{R}^d$ with $\dim(\operatorname{aff}(\sigma)) = d-1$. Let $\gamma\colon [0, 2\pi) \to \mathbb{S}^{d-1}$ be an angularly parameterized circle of directions so that $\gamma(0)$ is one of the two (antipodal) directions perpendicular to σ. For any point $p \in \mathbb{R}^d$, define the *γ-normal of p relative to σ* to be $\gamma(0)$ if the $\gamma(0)$-coordinate of p is at least that of σ, and $\gamma(\pi)$ otherwise. See Figure 1 (c).

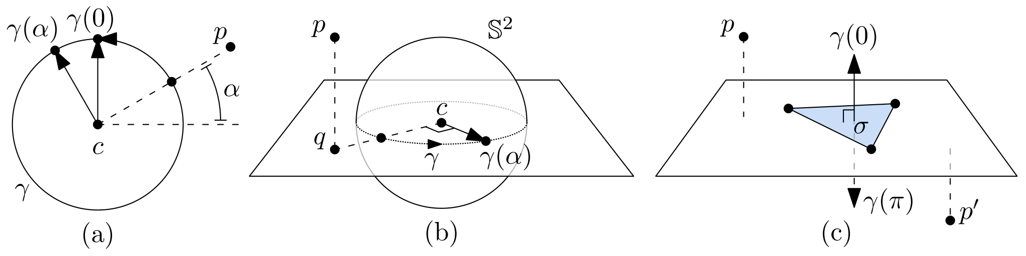


Figure 1 (a)-(b) $\gamma(\alpha)$ is the γ-normal of p. (c) since σ has codimension one with the ambient space, any point above (or below) σ has a γ-normal of $\gamma(0)$ (or $\gamma(\pi)$, respectively).

If $v_1 \neq v_2$ are vertices such that $\mathrm{aff}(\sigma \cup \{v_1\}) = \mathrm{aff}(\sigma \cup \{v_2\})$, but v_1 and v_2 are on "opposite sides" of σ, and if the γ-normal of v_1 relative to σ is $\gamma(\alpha)$, then the normal for v_2 is $-\gamma(\alpha)$.

3 Sweeping Orders

A main goal of this paper is to show how to compute an order on simplices (along with a corresponding list of directions perpendicular to the simplices) in order to emulate properties characteristic of sweepline algorithms with discrete points or vertices as events.

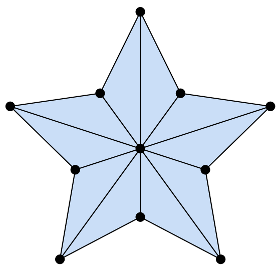


Figure 2 Here, every edge e has other edges in the halfspaces on either side of e. This general higher-dimensional phenomenon contrasts the special zero-dimensional case; in every direction, we can find at least one vertex with no other vertices below it. However, for each dimension of simplex, there *do* exist simplices with no *cofacets* in a halfspace below it, which we will see is true in general.

As illustrated in Figure 2, we cannot guarantee that all i-simplices below a particular i-simplex in our sweep appeared earlier in the order, so we instead focus on cofacets of simplices in the sweep. We introduce a sequence such that, for a given i-simplex σ and a corresponding direction $s \in \perp_\sigma$, all elements of $\mathrm{cof}_s^<(\sigma)$ are cofacets of some prior i-simplex in the sequence.

▶ **Definition 3.1** (Sweeping Order). A *sweeping order* of K_i is any sequence $SO_i := ((\sigma_j, s_j))_{j=1}^{n_i}$ that satisfies the following three properties.

1. Each s_j is a direction perpendicular to σ_j.
2. Each i-simplex of K appears exactly once in SO_i.
3. For any i-simplex σ_j, any cofacet in $\mathrm{cof}_{s_j}^<(\sigma_j)$ is also a cofacet of some σ_h, for $h < j$.

3.1 Computing a Sweeping Order

In this section, we show how to compute a sweeping order SO_i of K_i. For K_0, we simply pick an arbitrary direction s, order vertices by their s-coordinate (breaking ties arbitrarily), and output (v, s) for each vertex v in that order. See Algorithm 1.

We consider the $(i-1)$-simplices ρ in the order prescribed by a given sweeping order of K_{i-1}, denoted SO_{i-1}. First, suppose $\dim(\mathrm{aff}(\rho)) \leq d-2$. We radially order the cofacets σ of ρ that have not yet been output. Specifically, if SO_{i-1} pairs ρ with direction s, the radial order of its cofacets is based on a parameterized circle $\gamma_\rho \colon [0, 2\pi) \to \mathbb{S}^{d-1}$ of directions, rotating around ρ, starting at s. Each direction $\gamma_\rho(\alpha)$ corresponds to the unique halfspace whose boundary contains ρ, and whose exterior normal points in direction $\gamma_\rho(\alpha)$. For each cofacet $\sigma = \rho \cup \{v\}$ that has not yet been output, we intuitively consider the angle α_v for which σ enters this halfspace, i.e., the angle such that $\gamma_\rho(\alpha_v)$ is the γ_ρ-normal of v relative to ρ. We then output these cofacets $\rho \cup \{v\}$ in increasing order based on α_v, breaking ties arbitrarily, and pair them with the corresponding direction $\gamma_\rho(\alpha_v)$. Figure 3 illustrates the order in which various i-simplices are output.

Algorithm 1 Computing sweeping orders: SO_0, and SO_i given SO_{i-1} and K_i.

```
procedure ORDER(complex $K_0$)
    $s \leftarrow$ arbitrary direction
    for $v \in V(K_0)$ sorted increasingly by $s$-coordinate, breaking ties arbitrarily do
        output $(v, s)$

procedure ORDER(complex $K_i$, sweeping order $SO_{i-1}$ of $K_{i-1}$)
    for $(\rho, s)$ in $SO_{i-1}$ do        ▷ $s$ is a direction perpendicular to the $(i-1)$-simplex $\rho$
        $\gamma_\rho \leftarrow$ arbitrary circle $\gamma_\rho \colon [0, 2\pi) \to \mathbb{S}^{d-1}$ of directions maximally perpendicular to
                $\rho$, where the angle between $s$ and $\gamma_\rho(\alpha)$ is $\alpha$, so $\gamma_\rho(0) = s$ and $\gamma_\rho(\pi) = -s$
        $U_\rho \leftarrow \{$vertex $v$ of $K_0 \mid \rho \cup \{v\}$ is an $i$-simplex of $K_i$ that was not yet output$\}$
        for $v \in U_\rho$ do
            $\alpha_v \leftarrow \alpha$ such that $\gamma_\rho(\alpha)$ is the $\gamma_\rho$-normal to $v$ relative to $\rho$.
        for $v \in U_\rho$ sorted increasingly by $\alpha_v$, breaking ties arbitrarily do
            output $(\rho \cup \{v\}, \gamma_\rho(\alpha_v))$
```

If we encounter a simplex ρ whose affine hull has dimension $d-1$, there are only two (antipodal) directions perpendicular to it; in this case, we simply choose an $\mathbb{S}^1$ containing these two directions, so that all its cofaces have one of two possible angles. For clearer exposition, we encompass these two cases in the following definition.

▶ **Definition 3.2** (Maximally Perpendicular Circle). Let K be a simplicial complex in $\mathbb{R}^d$ and $\sigma \subseteq K$ be a simplex with $\dim(\text{aff}(\sigma)) < d$. We say that a circle of directions, γ_σ, is *maximally perpendicular (to σ)*, if 1) when $\dim(\text{aff}(\sigma)) < d-1$, we have $\gamma_\sigma \subseteq \perp_\sigma$, or 2) when $\dim(\text{aff}(\sigma)) = d-1$, we have $\perp_\sigma \subseteq \gamma_\sigma$.

That is, generally, γ_σ "rotates around" σ, except in the case that σ only has two directions perpendicular to it, in which case, these directions are contained in γ_σ. Since our method requires simplices to have directions perpendicular to them, we introduce Assumption 1, which we henceforth assume is satisfied by K.

▶ Assumption 1 (General Position for Enough Perpendiculars). Let K be a simplicial complex in $\mathbb{R}^d$. For every simplex $\sigma \subseteq K$, we require that $\dim(\text{aff}(\sigma)) \leq d-1$.

Note that this is quite a lenient condition, and does not prevent degeneracies. Proceeding with this assumption, $\text{ORDER}(K_i, SO_{i-1})$, Algorithm 1, takes as input K_i and SO_{i-1}, a sweeping order for K_{i-1}, and outputs a sequence of i-simplices and directions. The main result of this section is Theorem 3.5, which says $\text{ORDER}(K_i, SO_{i-1})$ is a sweeping order for K_i. We first show that it satisfies Properties 1 and 3 of Definition 3.1.

▶ **Lemma 3.3** (Directions are Perpendicular to their Paired Simplices). *Let $0 \leq i \leq \dim(K)-1$. If $i = 0$, let $SO_i = \text{ORDER}(K_i)$. If $i > 0$, let $SO_i = \text{ORDER}(K_i, SO_{i-1})$ for some sweeping order SO_{i-1}. For all elements $(\sigma, s_\sigma) \in SO_i$, the direction s is perpendicular to σ. That is, the output of Algorithm 1 satisfies Property 1 of Definition 3.1.*

Proof. Consider an arbitrary $(\sigma, s_\sigma) \in SO_i$. If $i = 0$, then σ is a vertex, and any direction is perpendicular to σ, including s. So consider the case $i > 0$. Then $\sigma = \rho \cup \{v\}$ for some $(i-1)$-simplex ρ and vertex v, where (ρ, s_ρ) is an index for the loop in Line 6, and v is an element of U_p (Line 9). On Line 7, we find the angle α_v such that $\gamma_\rho(\alpha_v)$ is the γ_ρ-normal of v relative to ρ. By Assumption 1, we have $\dim(\text{aff}(\rho)) \leq d-1$.

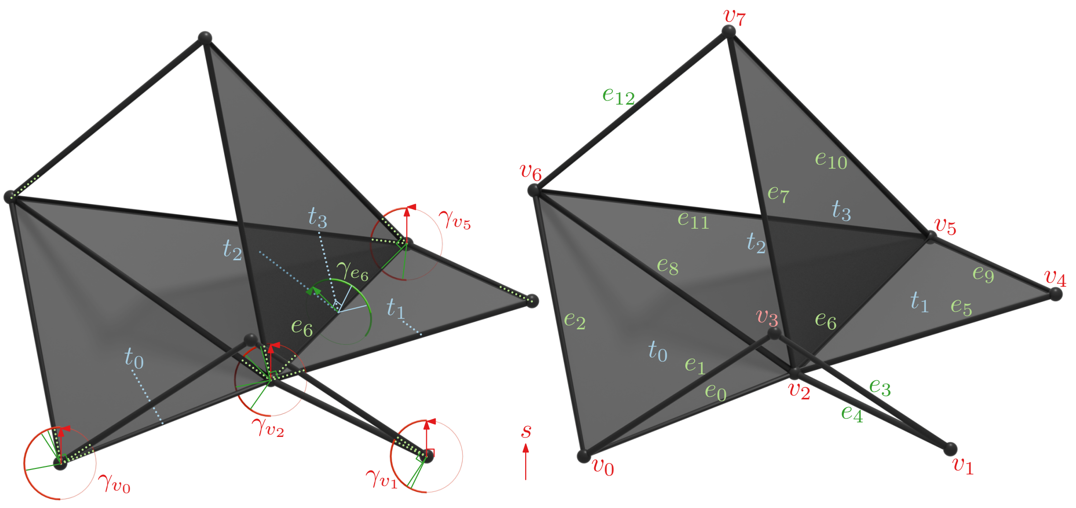


Figure 3 A simulation of $\text{Order}(K_0)$ with direction s, and $\text{Order}(K_i, SO_{i-1})$ for $i \in \{1,2\}$. Some circles γ_ρ relevant to the order are shown. Each γ_{v_i} lies in a plane parallel to the page. The circle γ_{e_6} lies in a plane perpendicular to edge e_6. A dotted line connecting a simplex σ to a facet ρ indicates that ρ is the facet that outputs σ. A solid line indicates the γ_ρ-normal with which some σ is output. Indices correspond to the order in which simplices are output.

If $\dim(\text{aff}(\rho)) < d-1$, then γ_ρ is normal to ρ, so the γ_ρ-normal $\gamma_\rho(\alpha_v)$ is well-defined and hence normal to σ. If $\dim(\text{aff}(\rho)) = d-1$, then we must have $\text{aff}(\sigma) = \text{aff}(\rho)$, otherwise $\dim(\text{aff}(\sigma))$ would be greater than $d-1$, violating Assumption 1. Then the γ_ρ-normal of v relative to ρ is $\gamma_\rho(0) = s_\rho$, which is perpendicular ρ and hence also to σ. ◀

▶ **Lemma 3.4** (Halfspace Property). *Suppose that* $\text{Order}(K_i, SO_{i-1}) = ((\sigma_j, s_j))_{j=1}^{n_i}$. *Then, for any* $1 \le j \le n_i$, *for each simplex* τ *of* $\text{cof}^<_{s_j}(\sigma_j)$, τ *is a cofacet of some* σ_h *with* $h < j$. *That is, Algorithm 1 satisfies Property 3 of Definition 3.1.*

Proof sketch. The case $i = 0$ is trivial. Suppose that $i > 0$ and $\text{Order}(K_i, SO_{i-1})$ outputs (σ_j, s_j) in iteration (ρ, s). Let v_j be the vertex such that $\sigma_j = \rho \cup \{v_j\}$. We show for any simplex $\tau = \sigma_j \cup \{v_h\} \in \text{cof}^<_{s_j}(\sigma_j)$, that the simplex $\sigma_h = \rho \cup \{v_h\}$ satisfies the claim.

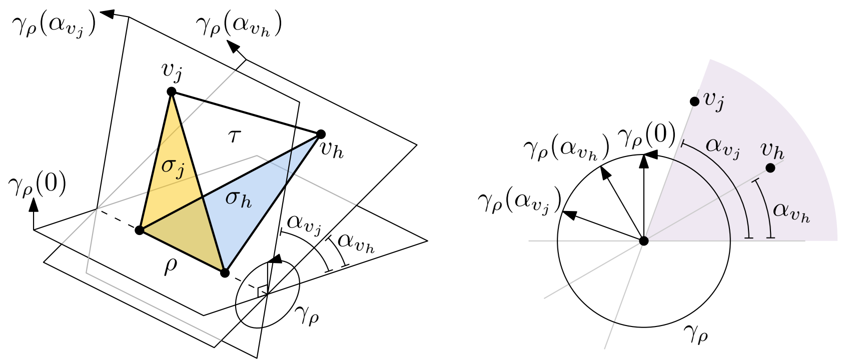


Figure 4 For some $\sigma_j = \rho \cup \{v_j\}$, we consider a cofacet $\tau = \rho \cup \{v_j, v_h\}$ (unshaded) for which the vertex v_h lies below σ_j with respect to the direction $\gamma_\rho(\alpha_{v_j})$. The simplex $\sigma_h = \rho \cup \{v_h\}$ is a cofacet of ρ that also has τ as a cofacet. The simplex ρ has a perpendicular circle of directions γ_ρ.

Consider the nontrivial case where σ_h is output in iteration (ρ, s) (and not before). Since

(ρ, s) is an element of a sweeping order for K_{i-1}, it satisfies Property 3 of Definition 3.1, so both σ_j and σ_h must be elements of $\mathrm{cof}_s^{\geq}(\rho)$, otherwise they would have been output in a previous iteration. Then $s{\cdot}v_j \geq s{\cdot}\rho$ and $s{\cdot}v_h \geq s{\cdot}\rho$, where $s = \gamma_\rho(0)$. Let $\gamma_\rho(\alpha_{v_j})$ and $\gamma_\rho(\alpha_{v_h})$ denote γ_ρ-normals of v_j and v_h relative to ρ, respectively. Because $\sigma_j \cup \{v_h\} \in \mathrm{cof}_{s_j}^{<}(\sigma_j)$, and since $s_j = \gamma_\rho(\alpha_{v_j})$, we have $\gamma_\rho(\alpha_{v_j}) \cdot v_h < \gamma_\rho(\alpha_{v_j}) \cdot \sigma_j$. Then v_h lies in the open halfspace containing σ_j in its boundary with exterior normal $\gamma_\rho(\alpha_{v_j})$, but not in the open halfspace containing ρ in its boundary with exterior normal $\gamma_\rho(0)$ (see the shaded sector in Figure 4). Then $0 \leq \alpha_h < \alpha_j$, so σ_h is output before σ_j on Line 11, and the claim is satisfied. ◀

Algorithm 1 satisfies Property 2 of Definition 3.1 trivially, so Theorem 3.5 follows.

▶ **Theorem 3.5.** *Order(K_i, SO_{i-1}), Algorithm 1, outputs a sweeping order for K_i in $O(n_{i-1} di \min\{d, i\} + n_i(i + d + \log n_0))$ time.*

References

1 Herbert Edelsbrunner and John L Harer. *Computational topology: an introduction.* American Mathematical Society, 2022.

2 Brittany Terese Fasy, Samuel Micka, David L Millman, Anna Schenfisch, and Lucy Williams. Efficient graph reconstruction and representation using augmented persistence diagrams. In *Canadian Conference on Computational Geometry*, 2022.

3 Allen Hatcher. *Algebraic Topology.* Cambridge Univ. Press, Cambridge, 2002.

Romeo and Juliet have a dog - Shortest paths to visibility for three agents inside simple polygons

Michelle Reimann[1] **and Fabian Stehn**[2]

1 **Technische Universität Berlin**
`m.reimann@campus.tu-berlin.de`

2 **Universität Bayreuth**
`fabian.stehn@uni-bayreuth.de`

Abstract

In this abstract, a generalized *watchman route problem* is studied: Given three agents at initial positions r, j, d in the interior of a simple polygon P. The task is to compute target positions (and shortest paths in P to) r', j', and d' in P so that the agent at j' can *see* the other agents that traveled to r' and d', respectively. The objective is to either minimize the sum of all shortest paths (MinSum) or the longest shortest path (MinMax). This extends to the two-agent case studied by Ahn et al. [1]. This initial work presents absolute error approximations for both variants.

1 Introduction

A problem related to the well-known *watchman route problem* [3, 4, 5, 6] is the so-called *quickest pair-visibility* problem with two agents in a polygon. The shortest paths for the agents within the polygon are to be determined so that the agents can see each other.

Related Work The body of research on *watchman route* problems and their variants is too rich and vast to be suitably presented in this extended abstract. We refer to the book by O'Rourke [13] for an overview of related problems.

In one flavor of this problem, one has to compute the shortest paths for a set of agents within a given domain to establish visibility. The objective of minimizing the sum of the path lengths is referred to as the MinSum variant, whereas computing a solution with the shortest longest path is called the MinMax variant.

Wynters and Mitchell [15] studied a setting for two agents acting in a polygonal domain with polygonal obstacles and provided solutions for the MinSum objective in $\mathcal{O}(En)$, and for the MinMax objective in $\mathcal{O}\left(n^3 \log n\right)$ time, respectively. Where E is the number of edges of the visibility graph of all vertices, and n is the total number of vertices of the obstacles.

For both variants Ahn et al. [1] provided linear time solutions considering two agents within a simple polygon with n corners without holes.

Also, query versions to this problem have been studied [1, 10, 11, 14]. Here, a polygon P and an initial position $s \in P$ are given. The aim is to preprocess the input to provide a structure that allows reporting the shortest path $d_P(s,t) \subseteq P$ so that visibility between t and a query point q is established. For simple polygons with n corners, a structure of size $\mathcal{O}(n)$ can be constructed in $\mathcal{O}(n)$ time, to answer queries in $\mathcal{O}(\log n)$ time [11]. Improved algorithms were developed by Arkin et al. [2] for the problem in the polygonal domain and by Wang [14] for polygons with a relatively small number of holes. For the MinMax variant in a simple polygon of n vertices where two initial positions are given, Ahn et al. [1] presented a solution with linear pre-processing time to report the shortest paths to a query point pair in $\mathcal{O}\left(\log^2 n\right)$ time.

However, to the best of our knowledge, this problem has not been studied for more than two agents.

Overview In this extended abstract, we study the problem of three agents inside a simple polygon of n vertices. The task is to compute the shortest paths from their initial positions to positions so that at least one agent sees the other two, resulting in a connected visibility graph. For the MinMax as well as the MinSum variant, we present absolute error approximation algorithms with a runtime in $\mathcal{O}\left(n^4 + n^2m^2\right)$ with m depending on the geometry of the polygon and given tolerance. An optimal solution for three or more agents remains open.

2 Preliminaries

Definitions and Notation Let P be a simple polygon and let $a, b \in P$ be two points in P (in the interior or on the boundary ∂P of P). We say that "*a and b see each other*" or $vis_P(a, b)$, iff $\overline{ab} \subseteq P$.

$LoS(\overline{ab})$ is the longest segment in P that contains a and b (with $vis_P(a, b)$) and is called a *line-of-sight (LoS)*. We denote the shortest path from a to b in P by $d_P(a, b) \subseteq P$ and its length by $|d_P(a, b)|$.

Problem Statement and Setup Various constraints can be applied to the visibility graph $G = (S^*, E)$ for multiple agents, where S^* represents their target positions. In this graph, an edge (s_i^*, s_j^*) is in E iff $vis_P(s_i^*, s_j^*)$. For instance, the graph G may represent a clique, where all nodes are mutually visible. Alternatively, it may contain a star graph with $|S^*| - 1$ spokes, where a central node is visible to all others. More generally, G may be a connected graph, ensuring that every node pair is visible through intermediate nodes. The latter case is the multiagent *quickest pair-visibility* problem studied here, which can be stated as follows.

Problem Distance-To-Relay Problem (DRP)
given: A simple polygon P with n vertices, a sequence $S = (s_1, \dots, s_k) \in P^k$ of points in P.
task: Compute $S^* = (s_1^*, ..., s_k^*) \in P^k$, so that there is a spanning tree $T = (S^*, E)$ with $E \subseteq P$ (the two endpoints of each edge see each other) minimizing either

$$\underbrace{\mu(S^*) := \max_{1 \le i \le k} |d_P(s_i, s_i^*)|}_{\text{MinMax}} \quad \text{or} \quad \underbrace{\mu(S^*) := \sum_{1 \le i \le k} |d_P(s_i, s_i^*)|}_{\text{MinSum}} \tag{1}$$

Depending on which of the two objectives μ of (1) is optimized, we refer to the MinMax or the MinSum variant, respectively.

For $k = 2$ DRP collapses to the "Romeo and Juliet" Problem studied in [1]. In this abstract, we restrict ourselves to $k = 3$: To Romeo ($r \in P$), Juliet ($j \in P$), and their dog ($d \in P$). We therefore assume that the layout of the network is known: The edges of the spanning tree are fixed to $\{r^*, j^*\}$ and $\{j^*, d^*\}$, see Fig. 1. For $k = 3$, this also corresponds to a star with 2 spokes. The agent initially positioned at j (Juliet) will also be called the "middle" agent and the other two agents will be called "outer" agents. For an optimal solution (r^*, j^*, d^*) it must hold that $vis_P(r^*, j^*)$ and $vis_P(j^*, d^*)$. The two optimal lines-of-sight are determined by $l^r := LoS(\overline{r^*j^*})$ and $l^d := LoS(\overline{j^*d^*})$.

Observations & Classification Suppose the optimal final position of the middle agent j^* is known. In that case, an optimal line-of-sight l^r is determined by the last two points of the shortest path $d_P(r, j^*)$. An optimal end position r^* lies on l^r and is the last point on the shortest path $d_P(r, l^r)$. The line-of-sight l^r is therefore tangential to a point on $d_P(r, j^*)$, which is a reflex corner of P [12], unless $vis_P(r, j^*)$ (analogous for l^d and d).

An instance of the DRP problem, independent of the objective function to be minimized, can be classified into the following types:

T1 (**trivial**), as the agents already see each other at their initial positions. The initial positions (r, j, d) are optimal destinations, consequently the objective function value is 0.

T2 ("**two agent reduceable**"), if an optimal solution (r^*, j^*, d^*) contains an optimal solution for two agents (w.l.o.g. r and j), see Fig. 1.

T3 ("**full**"), which is neither of type $T1$ nor $T2$, see Fig. 2.

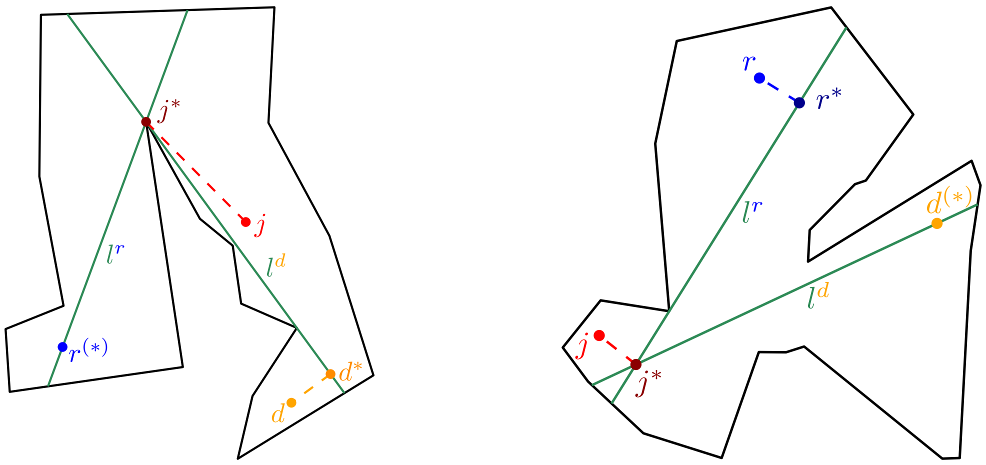


Figure 1 Instances of type $T2$ in the MinSum **(left)** and MinMax variant **(right)** containing an optimal solution (r^*, j^*) for the quickest pair-visibility problem with two agents.

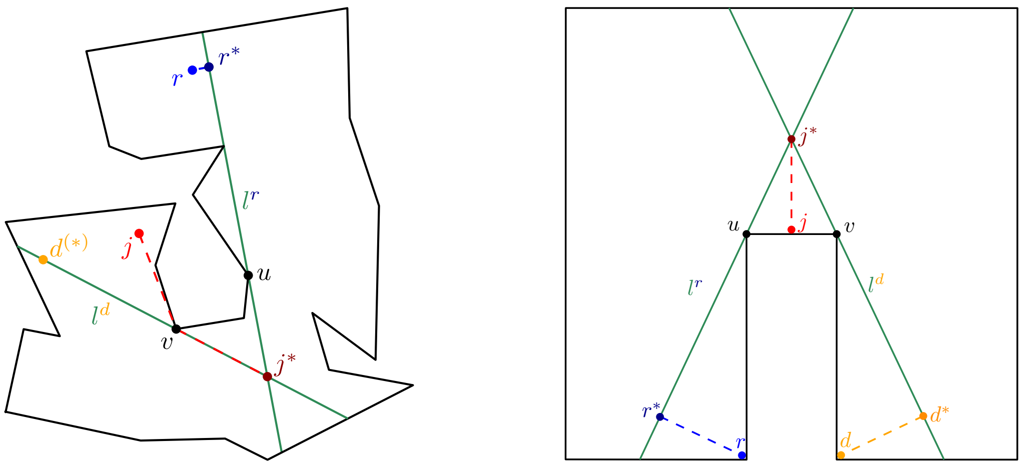


Figure 2 Instances of type $T3$ in the MinSum **(left)** and MinMax variant **(right)**.

$T1$ instances can be identified in linear time. Instances of type $T2$ can be solved in the following way: Fix and remove one (outer) agent s_a from the instance. Compute an optimal solution $opt_2 = (s_b^*, s_c^*)$ for the remaining agents s_b and s_c (using the linear time algorithm of [1]) and compute the shortest path length $d_a = d_P(s_a, s_a^*)$ so that s_a^* sees either s_b^* or s_c^* using [11]. If d_a is at most $\mu(opt_2)$ for the MinMax variant, a $T2$ instance is identified. For MinSum problems, a $T2$ instance might be at hand, even if $d_a > 0$: The contribution of s_a

might be too small to force a combinatorial change to opt_2. After $\mathcal{O}(n)$ preprocessing, shortest paths can be computed in $\mathcal{O}(\log n)$ [7, 8], therefore type $T2$ instances can be identified/tested as candidates in $\mathcal{O}(n)$ time.

In the following, we give the first results for instances of type $T3$. Here, all three agents influence the combinatorics of an optimal solution, preventing reductions to two-agent scenarios.

3 The Algorithm for T3 Instances

The value of the objective function for an optimal solution is derived from three values: Two distances of the outer agents (Romeo and the dog) to their respective optimal line-of-sight (l^r and l^d) and the distance of the middle agent (Juliet) to their intersection. Recall that in an optimal $T3$ solution, l^r and l^d have to be tangent to some reflex vertex of P.

The central idea of the additive error approximation is as follows: We *guess* the right pair (u and v) of reflex vertices of P, with $u \in l^r$ and $v \in l^d$. For u (v), we compute a set L_u (L_v) of candidate LoS. These segments will be placed in a way that guarantees that the length difference of the shortest paths from an outer agent to its optimal LoS and to the *next* candidate LoS is at most an absolute error term κ (whose value will be determined later).

Some candidate positions for the middle agent are crossings $l_u \cap l_v$ with $l_u \in L_u, l_v \in L_v$. By placing suitably distributed additional candidate positions for the middle agent on the segments of L_u and L_v, we ensure a candidate within distance κ to j^*.

The Set of Candidate Lines-Of-Sight Let $V(P)$ be the set of all corners of P, $\hat{V}(P)$ be all reflex vertices, and let $v \in \hat{V}(P)$. As shown in Fig. 3, the set $L_v = A_v \cup B_v$ consists of

1. lines-of-sight that contain v and a corner of $V(P)$ that is visible from v: $A_v := \{LoS(\overline{vv'}) \mid v' \in V(P), v \neq v', vis_P(v, v')\}$,
2. lines-of-sight that contain v and a point of S_κ (a set of $\lceil |\partial P|/\kappa \rceil$ points equally distributed along ∂P) visible from v: $B_v := \{LoS(\overline{vp}) \mid p \in S_\kappa, v \neq p, vis_P(v, p)\}$.

A_v ensures that an optimal LoS tangent to v can always be rotated (in either direction) until it coincides with an element of that set without swiping over a vertex of $V(P)$. The candidates of B_v ensure that the largest distance of any point on one candidate LoS (or between two consecutive LoS) to the next (previous) LoS (CW or CCW) is no more than κ.

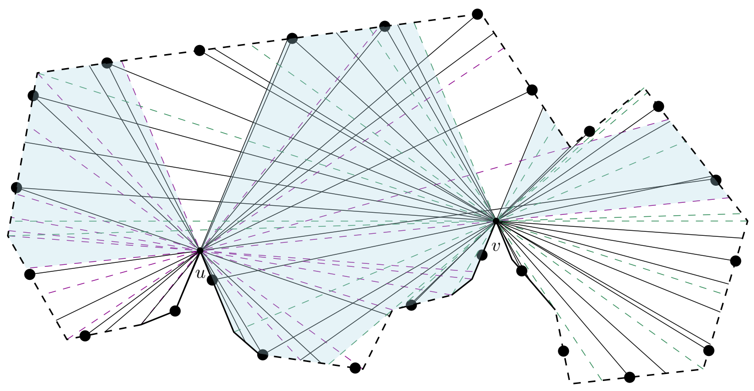


Figure 3 Common rotation area R (light blue) at u, v. $L_u = A_u$(purple) $\cup B_u$ and similarly, $L_v = A_v$(green) $\cup B_v$ are shown. Grid cells in R are delimited by consecutive LoS l_u, l'_u and l_v, l'_v.

Candidates Positions for the Middle Agent Let u, v be chosen so that $u \in l^r, v \in l^d$. Some candidate positions for the middle agent are crossings of the set $M_{uv} := \{l_u \cap l_v \mid l_u \in L_u, l_v \in L_v\}$. Consider the cells of the subdivision of P induced by the segments of M_{uv}. The optimal position j^* of the middle agent must be in one of these cells. However, the distance of j^* to its closest corner of that cell might be larger than κ. Therefore, we introduce additional candidate positions S_l along each segment $l \in L_u$ ($\in L_v$), as well as at the two endpoints of each segment, ensuring that the distance between consecutive candidates is at most $\sqrt{3}\kappa$, see Fig. 4. It is easy to see that the space between two consecutive LoS can be covered by discs of radius κ whose centers are placed at distance $\sqrt{3}\kappa$ along these lines-of-sight, as the endpoints of two consecutive LoS are at most κ apart. This, in turn, provides a covering for each cell of the subdivision by discs of radius κ placed on the boundary of that cell.

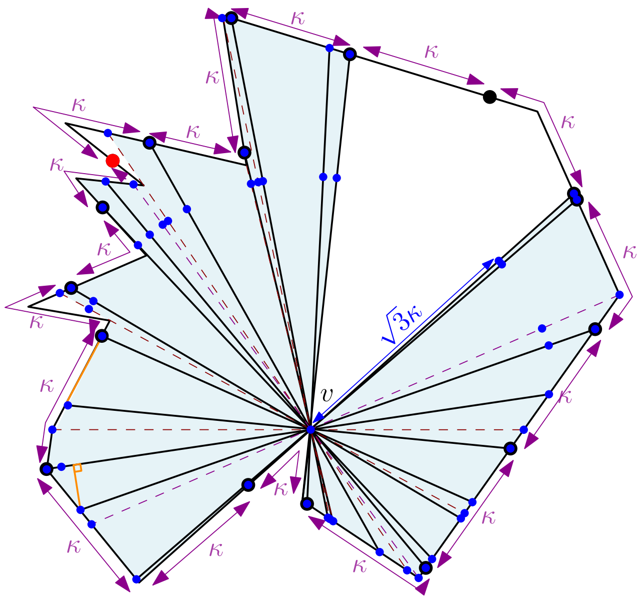


Figure 4 B_v after discretization of the rotation area (light blue) at v, S_k (black dots) and S_l (blue dots). Note that v cannot see the red point in S_k but combined with A_v (dashed) a maximal shortest path (orange) of length κ between consecutive LoS l_v, l'_v is ensured.

Computing the Approximation The approximation strategy is as follows: For each reflex vertex v of P compute the set L_v and for each segment of $l \in L_v$

- compute and store the values $l.r = d_P(r, l)$ and $l.d = d_P(d, l)$ in l,
- place candidate points S_l along l at distance $\sqrt{3}\kappa$ and at the endpoints of l.

Each candidate $\tilde{j} \in \bigcup_{u,v \in \hat{V}(P)} M_{uv}$ (later referred to as **type A**) is at the intersection of two LoS l_u and l_v and has an associated objective value of either $\max(d_P(j, \tilde{j}), l_u.r, l_v.d)$ or $d_P(j, \tilde{j}) + l_u.r + l_v.d$, respectively. The distances of all intersections on a fixed LoS to j can be computed in $\mathcal{O}(n + {}^{|\partial P|}/_{\kappa})$, as described later. Each candidate $\tilde{j} \in \bigcup_{\substack{u \in \hat{V}(P) \\ l \in L_u}} S_l$ (later referred to as **type B**) is placed on only one candidate LoS storing the contribution to the objective for one outer agent. For the other outer agent, the shortest path length from r to $\tilde{r}$ to establish $vis_P(\tilde{r}, \tilde{j})$ (d to $\tilde{d}$ to establish $vis_P(\tilde{d}, \tilde{j})$) can be computed using [11]. The objective value for these candidates can be derived as described above.

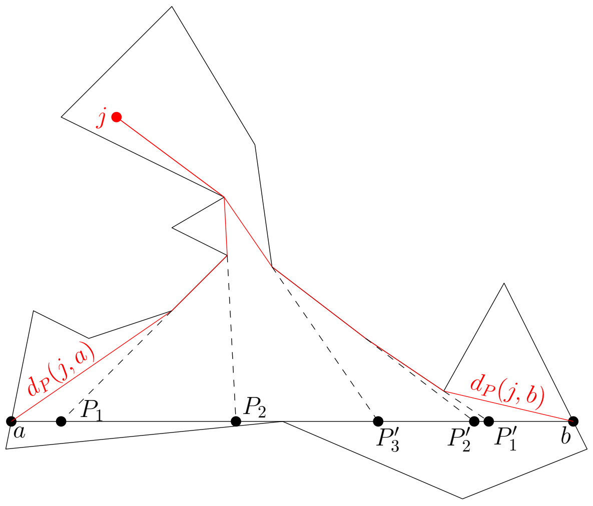


Figure 5 Funnel formed by $d_P(j,a)$ and $d_P(j,b)$ and cells with the combinatorially same shortest path to j.

Paths for Juliet to Intersections To compute the shortest paths $d_P(j,\tilde{j})$ for $\tilde{j} \in M_{uv}$, we fix a LoS $l = \overline{ab} \in L_u$ and consider the set $X = \{l \cap l' \mid l' \in L_v\}$ of intersections. The shortest paths $d_P(j,a)$ and $d_P(j,b)$ form a funnel. In the following, we focus on the parts of these paths that are not in the common prefix, as illustrated in Fig. 5, where paths may begin identically. For each pair of consecutive nodes p, q on the funnel, we intersect the ray from p through q with l. This yields points $P_1 \dots P_k$ ($P'_1 \dots P'_m$) for all pairs of consecutive nodes on the shortest path to a (b). Along l the points are ordered in the sequence $F = (P_1 \dots P_k, P'_m \dots P'_1)$, see Fig. 5. This step requires linear time and results in $\mathcal{O}(n)$ subdivisions of l. All points c on l that lie between two consecutive points in F have combinatorially the same shortest path to j: first to the corresponding node on the funnel border and then directly to c. By storing the length of the shortest path to j in the funnel nodes, we can easily add the distance from c to this funnel point. As we traverse from a to b, we consider all samples on l (from X and F). If we cross a point from F, we update the distance in the funnel node. For a sample from X, we calculate the shortest path in constant time per sample (the distance to the funnel node plus the stored value). Since both the samples and points in F are sorted, we can handle the next event in constant time. In total, there are $\mathcal{O}(n + |X|)$ events. Thus, for a fixed LoS l, we can compute the distances of all sample points from X in $\mathcal{O}(n + {}^{|\partial P|}/_{\kappa})$.

Analysis Every tested position $\tilde{j}$ gives rise to approximate solutions $(\tilde{r}, \tilde{j}, \tilde{d})$ that are valid, i.e., $vis_P(\tilde{r}, \tilde{j}) \wedge vis_P(\tilde{j}, \tilde{d})$. Let (r^*, j^*, d^*) be an optimal solution with an objective function value of *opt* and let $u, v \in \hat{V}(P)$ be the vertices at which l^r and l^d are tangent, see Fig. 6. By construction, we have that

1. $\tilde{j}$ is contained in or is on the boundary of a cell C^* induced by M_{uv} and P,
2. there is a candidate $\tilde{j} \in \partial C^*$ with $|\tilde{j} - j^*| \leq \kappa$.

Let $\tilde{l}^r$ ($\tilde{l}^d$) be the LoS containing $\tilde{j}$ and u ($\tilde{j}$ and v). Again by construction we have that $|d_P(r, \tilde{l}^r) - d_P(r, l^r)| \leq \kappa$ and $|d_P(d, \tilde{l}^d) - d_P(d, l^d)| \leq \kappa$. Therefore choosing $\kappa = \varepsilon$ for the MinMax or $\kappa = {}^{\varepsilon}/_{3}$ for the MinSum variant establishes an absolut error approximation with $app \leq opt + \varepsilon$, where *app* is the smallest objective function value for any candidate $\tilde{j}$.

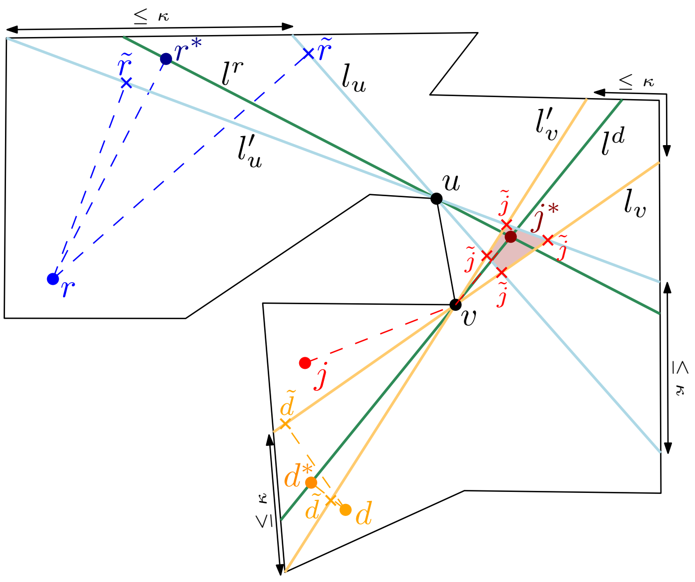


Figure 6 Optimal solution (r^*, j^*, d^*) compared to approximated solutions $(\tilde{r}, \tilde{j}, \tilde{d})$.

Runtime With linear preprocessing time, the shortest path lengths between two arbitrary points in P can be computed in $\mathcal{O}(\log n)$ [7, 8] as well as the shortest path length from a source point to *see* a query point [11, 2].

The size of L_v for a vertex $v \in \hat{V}(P)$ is bounded by $\mathcal{O}\left(n + |\partial P|/\kappa\right)$. By introducing the vertices of S_κ as additional vertices on P, all LoS of L_v can be constructed as a byproduct of the linear time algorithm [9] to compute the visibility polygon of v in P. Along each line-of-sight $l \in L_v$ additional $\mathcal{O}\left(|\partial P|/\kappa\right)$ candidate positions (S_l) are introduced.

For $\tilde{j}$, there are $\mathcal{O}\left(\binom{|\hat{V}(P)|}{2}\left(n + |\partial P|/\kappa\right)^2\right)$ candidates of type A that can be evaluated in constant time and $\mathcal{O}\left(|\hat{V}(P)|\left(n + |\partial P|/\kappa\right)\left(|\partial P|/2\sqrt{3}\kappa\right)\right)$ candidates of type B that can be evaluated in $\mathcal{O}(\log n)$ time. This results in a combined runtime of $\mathcal{O}\left(n^2 \cdot \left(n + |\partial P|/\varepsilon\right)^2\right)$.

4 Conclusion

In this extended abstract we presented an absolute error approximation algorithm $(opt + \varepsilon)$ with a runtime of $\mathcal{O}\left(n^2 \cdot \left(n + |\partial P|/\varepsilon\right)^2\right)$ for a polygon P with n vertices, for any $\varepsilon > 0$.

Several questions remain open and qualify as starting points for further research projects, such as finding an optimal solution to this problem or generalizing the results to an arbitrary number of agents. Also, instead of computing a "visibility tree", one might be interested in computing solutions where all agents can see each other or extending the problem setting to polygons with holes.

References

1 Hee-Kap Ahn, Eunjin Oh, Lena Schlipf, Fabian Stehn, and Darren Strash. On romeo and juliet problems: Minimizing distance-to-sight. *Computational Geometry*, 84:12–21, 2019. Special Issue on the 34th European Workshop on Computational Geometry. doi:10.1016/j.comgeo.2019.07.003.

2 Esther M. Arkin, Alon Efrat, Christian Knauer, Joseph SB Mitchell, Valentin Polishchuk, Günter Rote, Lena Schlipf, and Topi Talvitie. Shortest Path to a Segment and Quickest Visibility Queries. *Journal of Computational Geometry*, 7(2):77–100, January 2016. doi:10.20382/jocg.v7i2a5.

3 W Chin and S Ntafos. Optimum Watchman Routes. In *Proceedings of the Second Annual Symposium on Computational Geometry*, SCG '86, pages 24–33, New York, NY, USA, 1986. Association for Computing Machinery. doi:10.1145/10515.10518.

4 Wei-pang Chin and Simeon Ntafos. Optimum Watchman Routes. *Information Processing Letters*, 28(1):39–44, May 1988. doi:10.1016/0020-0190(88)90141-X.

5 Moshe Dror, Alon Efrat, Anna Lubiw, and Joseph Mitchell. Touring a Sequence of Polygons. *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*, April 2003. doi:10.1145/780542.780612.

6 Adrian Dumitrescu and Csaba D. Tóth. Watchman Tours for Polygons with Holes. *Computational Geometry*, 45(7):326–333, August 2012. doi:10.1016/j.comgeo.2012.02.001.

7 Leonidas J. Guibas and John Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126–152, 1989. doi:10.1016/0022-0000(89)90041-X.

8 John Hershberger. A new data structure for shortest path queries in a simple polygon. *Inf. Process. Lett.*, 38(5):231–235, June 1991. doi:10.1016/0020-0190(91)90064-O.

9 B. Joe and R. B. Simpson. Corrections to Lee's visibility polygon algorithm. *BIT Numerical Mathematics*, 27(4):458–473, 1987. doi:10.1007/BF01937271.

10 Ramtin Khosravi and Mohammad Ghodsi. The Fastest Way to View a Query Point in Simple Polygons. In *European Workshop on Computational Geometry*, pages 187–190, January 2005. URL: https://api.semanticscholar.org/CorpusID:2345696.

11 Christian Knauer, Günter Rote, and Lena Schlipf. Shortest inspection-path queries in simple polygons. In *Abstracts of the 24th European Workshop on Computational Geometry*, pages 153–156, 2008. URL: http://page.mi.fu-berlin.de/rote/Papers/pdf/Shortest+inspection-path+queries+in+simple+polygons.pdf.

12 Wolfgang Mulzer. Shortest Paths in Polygons. n.d. (Retrieved on May 15, 2024.). URL: https://page.mi.fu-berlin.de/mulzer/notes/alggeo/polySP.pdf.

13 Joseph O'Rourke. *Art Gallery Theorems and Algorithms.* Number 3 in The International Series of Monographs on Computer Science. Oxford University Press, New York, 1987.

14 Haitao Wang. Quickest Visibility Queries in Polygonal Domains. In *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 61:1–61:16, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2017.61.

15 Erik L Wynters and Joseph SB Mitchell. Shortest Paths for a Two-Robot Rendez-Vous. In *CCCG*, pages 216–221, 1993. URL: https://www.researchgate.net/publication/220991707_Shortest_Paths_for_a_Two-robot_Rendez-vous.

Flipping Matchings is Hard[*]

Carla Binucci[1], Fabrizio Montecchiani[1], Daniel Perz[1], and Alessandra Tappini[1]

1 **University of Perugia, Italy**
`carla.binucci|fabrizio.montecchiani|daniel.perz|alessandra.tappini@unipg.it`

Abstract

Given a point set $\mathcal{P}$ and a plane perfect matching $\mathcal{M}$ on $\mathcal{P}$, a flip is an operation that replaces two edges of $\mathcal{M}$ such that another plane perfect matching on $\mathcal{P}$ is obtained. Given two plane perfect matchings on $\mathcal{P}$, we show that it is NP-hard to minimize the number of flips that are needed to transform one matching into the other.

1 Introduction

A *straight-line drawing* Γ of a graph G on a point set $\mathcal{P}$ maps each vertex v of G to a distinct point p_v of $\mathcal{P}$ and each edge (u, v) of G to the straight-line segment $p_u p_v$. Such a drawing is *planar* if no two edges share a point, except at common endpoints. In what follows, we always refer to planar straight-line drawings of graphs.

Given a point set $\mathcal{P}$ of n points in the plane and a family $\mathcal{G}$ of drawings, an *edge flip* is the operation of replacing an edge of a drawing Γ in $\mathcal{G}$ with a different edge such that the resulting drawing is still in $\mathcal{G}$. The *flip graph* of $\mathcal{G}$ is a graph that has a vertex for every element of $\mathcal{G}$ and an edge between two vertices if their corresponding drawings differ by an edge flip. The main questions about flip graphs are their connectedness, their diameter, and the complexity of finding the shortest path between two vertices (i.e., the shortest sequence of edge flips that transforms one drawing into another). The connectedness and the diameter have been studied for different families of planar straight-line graph drawings like triangulations [19, 22, 28], spanning trees [1, 8, 11, 12, 13, 17, 25], spanning paths [4, 7, 14, 21], and odd matchings [3]. Regarding the complexity question, it has been shown that finding the shortest flip sequence between triangulations is not only NP-hard [5, 23], but also APX-hard [26]. On the positive side, there exists an FPT algorithm which uses the length of the flip sequence as a parameter [20].

For simplicity, in what follows we use *plane perfect matching* as a shorthand for planar straight-line drawing of a perfect matching. Note that, for plane perfect matchings, it is necessary to replace at least two edges at the same time to transform one matching into another. In this paper, we refer to this operation as a *flip*, and use the according definition of flip graph. It has been shown, if the point set is convex, then the flip graph of plane perfect matchings is connected and the shortest flip sequence between two given matchings is computable in polynomial time [18]. On the other hand, for point sets in general position, whether the flip graph is connected or not is still an open question. Further research focuses on non-plane perfect matchings [9, 15] or other variants of flip operations [2, 6, 24].

[*] Work of CB and AT supported in part by MUR PRIN Proj. 2022TS4Y3N - "EXPAND: scalable algorithms for EXPloratory Analyses of heterogeneous and dynamic Networked Data". Work of FM and DP supported in part by MUR PRIN Proj. 2022ME9Z78 - "NextGRAAL: Next-generation algorithms for constrained GRAph visuALization".

In this paper, we study the problem of deciding whether a plane perfect matching can be transformed into another with at most k flips, called FlippingBetweenMatchings:

FlippingBetweenMatchings
Input: $\langle \mathcal{P}, \mathcal{M}_1, \mathcal{M}_2, k \rangle$. A point set $\mathcal{P}$, two plane perfect matchings $\mathcal{M}_1$ and $\mathcal{M}_2$ on $\mathcal{P}$, and a positive integer k.
Question: Does there exist a sequence of at most k flips which transforms $\mathcal{M}_1$ into $\mathcal{M}_2$?

In [18], it is shown that FlippingBetweenMatchings can be solved in linear time if $\mathcal{P}$ is convex, and the authors leave it as an open problem to study the case in which $\mathcal{P}$ is not convex.

We solve this problem by proving that FlippingBetweenMatchings is NP-hard, even under the additional restriction that the points of $\mathcal{P}$ have integer coordinates and the area of the minimum-size axis-aligned bounding box containing $\mathcal{P}$ is polynomial in the size of the matching. Our contribution is summarized by the following theorem.

▶ **Theorem 1.1.** *FlippingBetweenMatchings is NP-complete, even for integer point sets whose area is polynomial in the size of the matching.*

Due to space limitations, we only describe the main ideas of the reduction; the complete proof is in the full version, see [10].

2 Proof of Theorem 1.1

Let $\mathcal{M}$ be a plane perfect matching on a point set $\mathcal{P}$. Let e_1 and e_2 be two edges of $\mathcal{M}$ whose endpoints are mapped on a subset $\mathcal{Q}$ of $\mathcal{P}$. A *flip of* e_1 *and* e_2 is an operation that eliminates e_1 and e_2 and introduces e_3 and e_4 such that their endpoints are still mapped on $\mathcal{Q}$, no edge crossing is introduced, and no two vertices are mapped to the same point. In other words, a flip operation produces a different plane perfect matching on $\mathcal{P}$; see Figure 1 in which $\mathcal{Q} = \{p_1, p_2, q_1, q_2\}$. In what follows, we may use the term *flip* without specifying the involved edges if they are clear from the context, or if we refer to a sequence of flips.

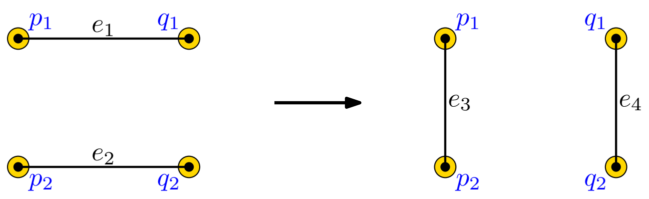


Figure 1 Flipping edges e_1 and e_2 to e_3 and e_4; points are drawn as yellow circles.

The membership of FlippingBetweenMatchings to NP is trivial, as one can guess all possible sequences of k flips, and, for each of them, verify whether it transforms $\mathcal{M}_1$ into $\mathcal{M}_2$. To prove hardness, we reduce from the NP-complete problem PlanarVertexCover [16]:

PlanarVertexCover
Input: $\langle G = (V, E), c \rangle$. A planar graph $G = (V, E)$, and a positive integer c.
Question: Does there exist a set of vertices $V_C \subseteq V$, called *vertex cover* of G, such that $|V_C| \leq c$ and every edge of G has at least one vertex in V_C?

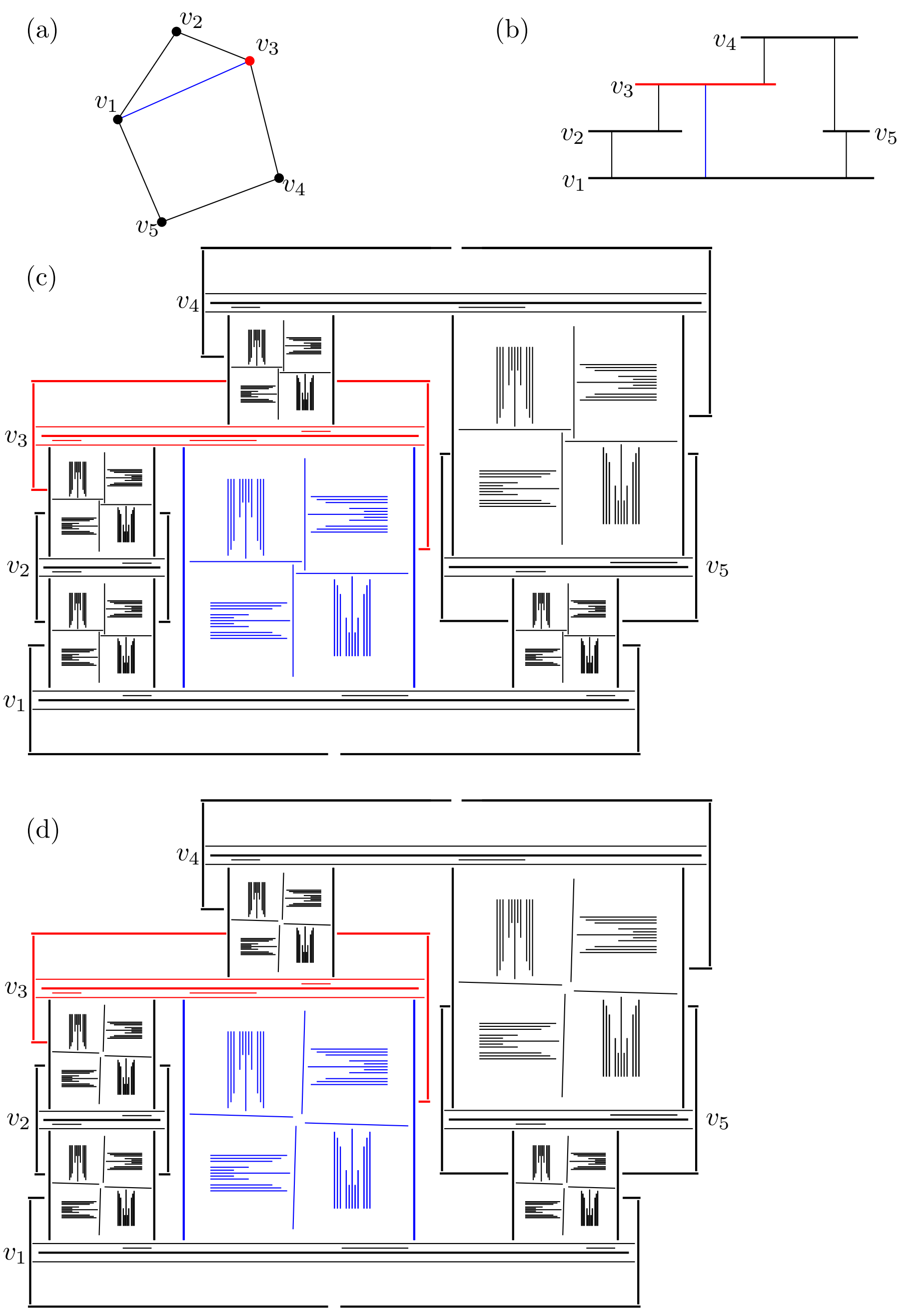


Figure 2 (a) A graph G where a vertex cover of size 3 is depicted in gray. (b) A weak-visibility representation R of G. (c) The plane perfect matching $\mathcal{M}_1$ obtained by replacing each vertex-segment of R by a vertex gadget and each edge-segment of R by an edge gadget. (d) The plane perfect matching $\mathcal{M}_2$. Each bold line represents $2k + 2$ line segments. The vertex v_3 is red and the edge v_1v_3 is blue in each representation.

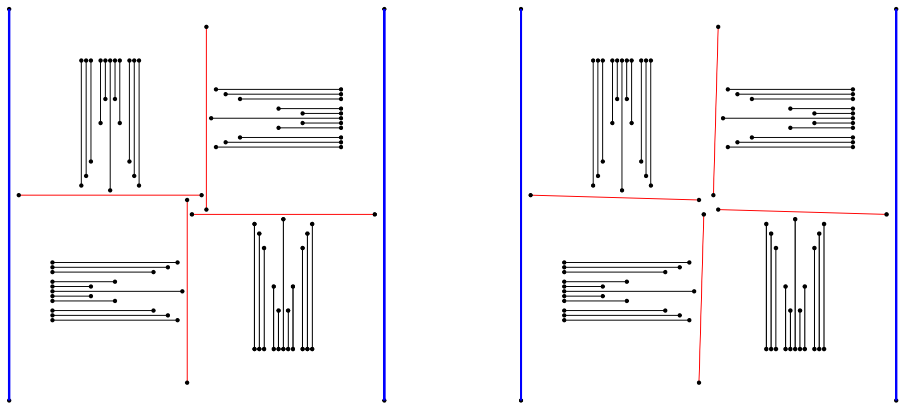

(a) Edge gadget in the start-configuration. **(b)** Edge gadget in the final-configuration.

Figure 3 Construction of an edge gadget. The edges of the flip structure are colored red, the edges of the blockers are colored black, and the edges of the separators are represented as a blue line segment.

Construction.

We begin by describing the construction behind our proof. Given an instance $I = \langle G = (V, E), c\rangle$ of PLANARVERTEXCOVER, we construct an instance $I' = \langle \mathcal{P}, \mathcal{M}_1, \mathcal{M}_2, k\rangle$ of FLIPPINGBETWEENMATCHINGS, with $k = 2c + 5|E|$, as follows; refer to Figure 2. At high level, in order to obtain $\mathcal{M}_1$, the vertices and edges of G are replaced by vertex gadgets and edge gadgets. Also, $\mathcal{M}_1$ and $\mathcal{M}_2$ only differ in the edge gadgets. Further, the fastest way to reconfigure an edge gadget should be through a constant-length flip sequence involving an edge of a vertex gadget. Our construction goes through a preliminary representation R in which all vertices (edges) of G are drawn as horizontal (vertical) segments.

From G to R. Since G is planar, it admits a weak-visibility representation [27], namely, a representation of G such that: each vertex is represented as a horizontal segment, each edge is represented as a vertical segment whose endpoints, called *attachments*, lie on the horizontal segments of the corresponding endpoints, no two segments intersect each other except possibly at attachment points, all endpoints of horizontal and vertical segments are on an integer grid of quadratic size in the number of vertices. (The representation is called weak because two horizontal segments may see each other even when the corresponding vertices are not adjacent in the graph.) Figure 2a shows a graph G with a vertex cover of size three depicted in gray, and Figure 2b shows a weak-visibility representation R of G.

From R to $\mathcal{M}_1$. Given a weak-visibility representation R of G, to obtain a plane perfect matching $\mathcal{M}_1$ and to define the corresponding point set $\mathcal{P}$, we replace each vertex-segment of R by a *vertex gadget* and each edge-segment of R by an *edge gadget*; Figure 2c shows the drawing $\mathcal{M}_1$ obtained from the weak-visibility representation R of Figure 2b.

An edge gadget consists of: (i) The *flip structure*, which contains four edges each having one endpoint in the central part of the gadget; (ii) four *blockers*, each consisting of eleven parallel edges; (iii) two *separators*, each consisting of $2k+2$ parallel edges. Refer to Figure 3 where the edges of a separator are represented as bold blue line segments. Intuitively, blockers prevent any two edges of the flip structure to be flipped within the same operation, while separators prevent any interplay between different edge gadgets.

A vertex gadget associated with a vertex-segment of R representing a vertex of G consists of three parts: One *frame*, depicted in red in Figure 4a, and two *vertex separators*, depicted in blue in Figure 4a, one for each of the two sides of the frame. The frame consists of several horizontal line segments: (i) The *top-edge* and the *bottom-edge*, which are the topmost edge and the bottommost edge, respectively, and they are the longest ones; (ii) the $2k+2$ *middle-edges*, which are shorter than both the top-edge and the bottom-edge; (iii) the *connectors*, which are edges that lie in the region between the top-edge and the middle-edges or in the region between the bottom-edge and the middle-edges. Each vertex separator consists of $2k+2$ vertical edges next to the frame and $2k+2$ horizontal edges above and $2k+2$ horizontal edges below these vertical edges. Figure 4 shows an example of a vertex gadget incident to three edge gadgets. Intuitively, for every incident edge gadget, a connector is placed such that, after flipping the top-edge and the bottom-edge, the connector can be flipped with an edge of the flip structure of the edge gadget. Vertex separators and middle-edges prevent any interplay between different vertex gadgets.

In what follows, we consider a particular flip operation, shown in Figure 4, which transforms a vertex gadget from a *deactivated* configuration (see Figure 4a) to an *activated* configuration (see Figure 4b). When performing this flip, we also say that we *activate* a vertex gadget and, conversely, we *deactivate* a vertex gadget if we perform the reverse operation.

Observe that to obtain $\mathcal{M}_1$ we need to "stretch" the visibility representation R, which corresponds to the introduction of additional rows and columns in the underlying grid (which is always possible if the inserted row or column only crosses vertical or horizontal segments, respectively); refer to Figure 2c for an example. More precisely, given two edges (u, v) and (u, w) of G such that the edge-segment representing (u, v) is longer than the edge-segment representing (u, w) in R, the corresponding edge gadgets have different sizes in $\mathcal{M}_1$ and, consequently, the vertex gadgets may need to have different lengths.

From $\mathcal{M}_1$ to $\mathcal{M}_2$. The plane perfect matching $\mathcal{M}_2$ differs from $\mathcal{M}_1$ only in the edge gadgets. More precisely, an edge gadget can assume two configurations, which we call the *start-configuration* and the *final-configuration*, based on the mapping of the four edges of the flip structure. Figure 3a and Figure 3b show the start-configuration and the final-configuration of an edge gadget, respectively. In $\mathcal{M}_1$ all the edge gadgets are in the start-configuration, whereas in $\mathcal{M}_2$ all the edge gadgets are in the final-configuration. A start-configuration can be transformed into a final-configuration by a sequence of five flips. Such a sequence of flips transforms an edge gadget associated with an edge e of G by using a connector of a vertex gadget that is associated with one of the endpoints of e. Observe that it is possible to use a connector of a vertex gadget only if the vertex gadget is activated. Figure 5 shows the sequence of five flips to transform an edge gadget from its start-configuration to its final-configuration.

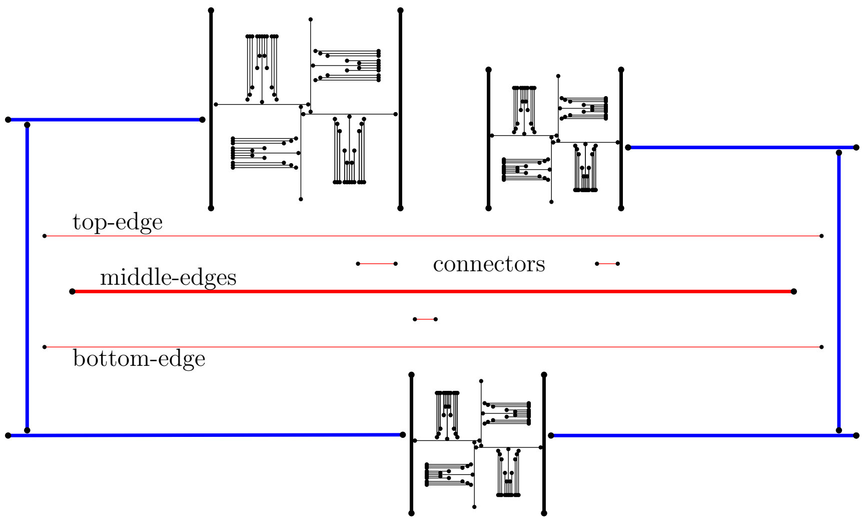
The point set $\mathcal{P}$. The described point set is of polynomial size, namely $|\mathcal{P}| \in O(|V|^2+|E|^2)$. Also, it is an integer point set, because it is constructed starting from a weak-visibility representation on an integer grid by introducing additional rows and columns when needed. However, the exact placement of the points depend on the length of the segments representing the edges of the matching, which are discussed in the full version and yield polynomial area.

Correctness.

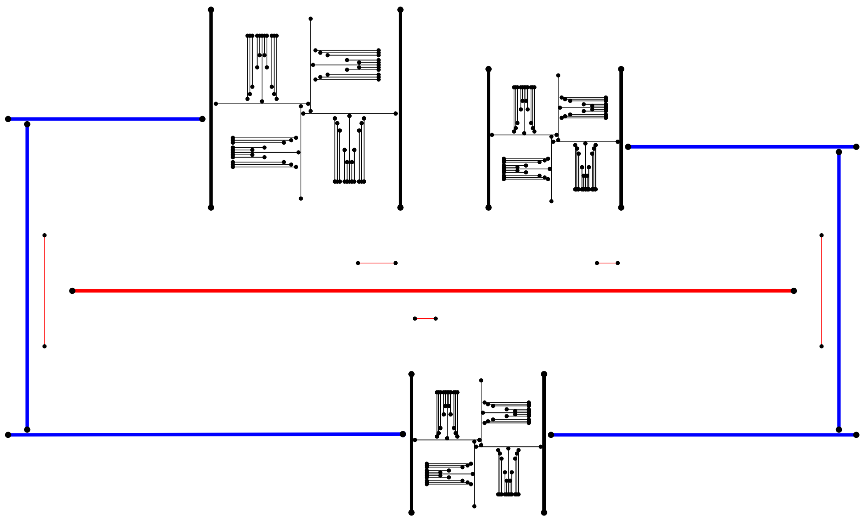
We now show the correctness of our reduction in the following two lemmas.

▶ **Lemma 2.1.** *If $I = \langle G = (V, E), c\rangle$ is a* `yes` *instance of* PlanarVertexCover *then $I' = \langle \mathcal{P}, \mathcal{M}_1, \mathcal{M}_2, 2c + 5|E|\rangle$ is a* `yes` *instance of* FlippingBetweenMatchings*.*

Proof. Let $V_C \subseteq V$ be a vertex cover of G of size $c' \le c$. We consider the c' vertex-segments of R corresponding to the vertices of V_C and we activate each of the corresponding vertex



(a) Deactivated vertex gadget. The bold edges are a set of $2k+2$ edges.



(b) Activated vertex gadget.

Figure 4 Two notable configurations of a vertex gadget.

gadgets of $\mathcal{M}_1$. For each edge-segment of R incident to these vertex-segments, we transform the corresponding edge gadget of $\mathcal{M}_1$ from the start-configuration to the final-configuration, as shown before. Once this has been done for each edge gadget, we deactivate the vertex gadgets that we previously activated, which yields to $\mathcal{M}_2$. The process requires exactly $2c' + 5|E|$ flips in total. Indeed, (i) for each edge gadget of $\mathcal{M}_1$, exactly 5 flip operations are required to transform the start-configuration into the final-configuration; (ii) to do these transformations, exactly c' vertex gadgets need to be activated and deactivated. ◀

▶ **Lemma 2.2.** *If $I' = \langle S, \mathcal{M}_1, \mathcal{M}_2, 2c+5|E|\rangle$ is a* `yes` *instance of* FLIPPINGBETWEENMATCHINGS *then $I = \langle G = (V,E), c\rangle$ is a* `yes` *instance of* PLANARVERTEXCOVER.

Proof sketch. We argue that the only way to transform $\mathcal{M}_1$ into $\mathcal{M}_2$ with a sequence of $2c+5|E|$ flips is through the activation of at most c vertex gadgets such that all edges of G have at least one vertex among the corresponding vertices. In other words, a feasible

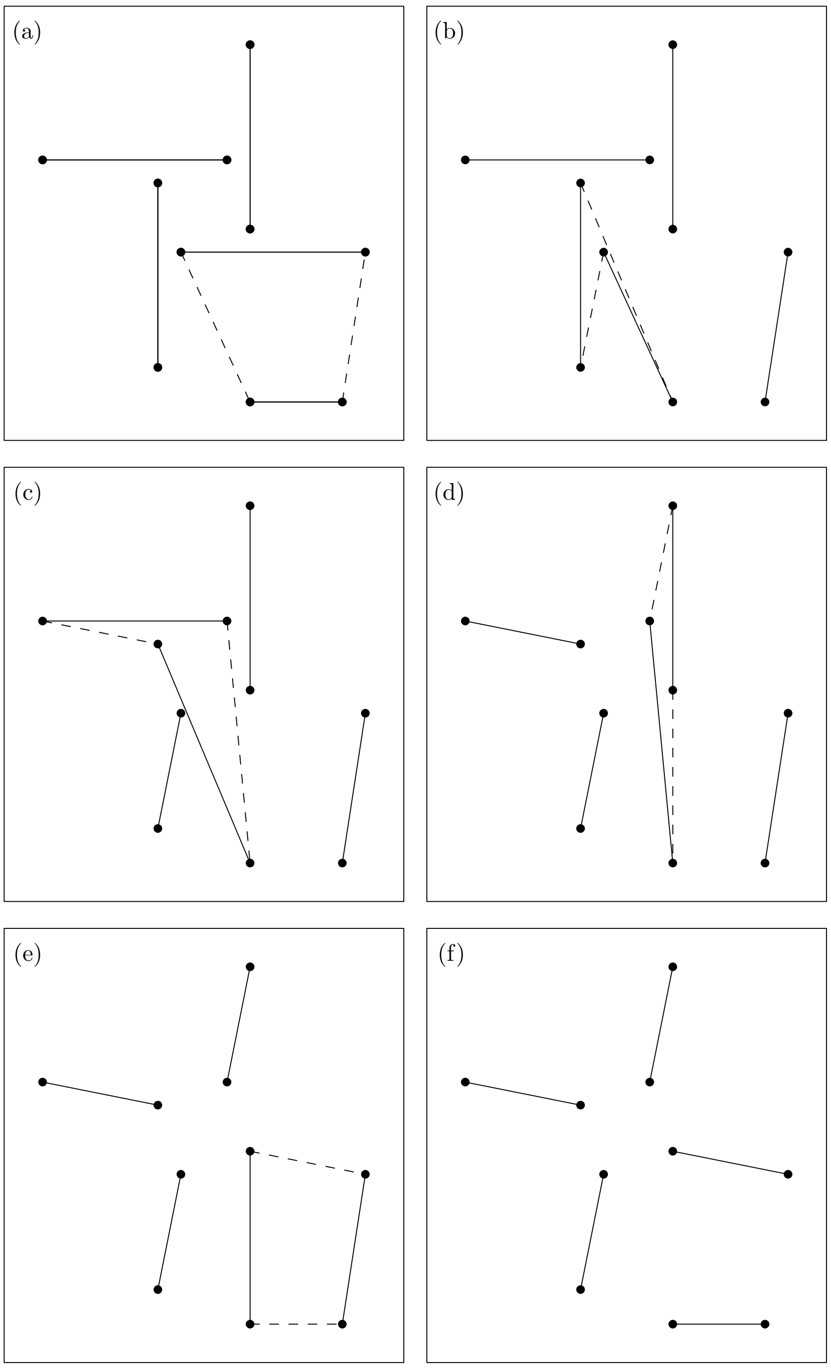


Figure 5 (a)–(e) A sequence of 5 flips to transform an edge gadget from the start-configuration to the final-configuration. (f) The final-configuration. In all subfigures, dashed edges are the ones that are added with the flip operation, and the bottommost edge in (a) and (f) is a connector of the vertex gadget. For the sake of readability, we only illustrate the flip structure of the edge gadget and the connector of the vertex gadget that is used to perform the transformation.

sequence of flips must identify at most c vertices forming a vertex cover of G. Due to the presence of the blocking structures and the separators for the edge gadgets and the vertex separators for the vertex gadget, we cannot transform $\mathcal{M}_1$ to $\mathcal{M}_2$ with a sequence of less than $2c + 5|E|$ flips. The proof is rather technical and completely deferred to the full version. ◀

3 Conclusion and Further Research

We have shown that deciding whether k flips suffice to transform a given plane perfect matching into another on the same point set is NP-complete. While the point set exploited in our reduction has integer coordinates and occupies polynomial area, it is not in general position. Therefore, a natural question is whether Theorem 1.1 holds for point sets in general position.

Recall that the connectedness of the flip graph of perfect matchings is still open. Toward solving this question, we are currently working on the setting in which our point set consists of at most two convex layers. Moreover, we are working on extending our NP-hardness proof for plane odd matchings and plane spanning paths.

References

1 Oswin Aichholzer, Brad Ballinger, Therese Biedl, Mirela Damian, Erik D. Demaine, Adam Hesterberg, Matias Korman, Anna Lubiw, Jayson Lynch, Josef Tkadlec, and Yushi Uno. Reconfiguration of non-crossing spanning trees. *Journal of Computational Geometry*, 15(1):224–253, 2024. doi:10.20382/jocg.v15i1a9.

2 Oswin Aichholzer, Sergey Bereg, Adrian Dumitrescu, Alfredo García, Clemens Huemer, Ferran Hurtado, Mikio Kano, Alberto Márquez, David Rappaport, Shakhar Smorodinsky, Diane Souvaine, Jorge Urrutia, and David Wood. Compatible geometric matchings. *Computational Geometry*, 42(6-7):617–626, 2009. doi:10.1016/j.comgeo.2008.12.005.

3 Oswin Aichholzer, Anna Brötzner, Daniel Perz, and Patrick Schinder. Flips in Odd Matchings. In *Proceedings of the 36th Canadian Conference on Computational Geometry 2024*, pages 303–307, St. Catharines, Canada, 2024. URL: https://cosc.brocku.ca/~rnishat/CCCG_2024_proceedings.pdf#page=313.

4 Oswin Aichholzer, Kristin Knorr, Wolfgang Mulzer, Johannes Obenaus, Rosna Paul, and Birgit Vogtenhuber. Flipping plane spanning paths. In *WALCOM: Algorithms and Computation*, pages 49–60. Springer, 2023. doi:10.1007/978-3-031-27051-2_5.

5 Oswin Aichholzer, Wolfgang Mulzer, and Alexander Pilz. Flip Distance Between Triangulations of a Simple Polygon is NP-Complete. *Discrete & Computational Geometry*, 54(2):368–389, 2015. doi:10.1007/s00454-015-9709-7.

6 Oswin Aichholzer, Julia Obmann, Pavel Paták, Daniel Perz, Josef Tkadlec, and Birgit Vogtenhuber. Disjoint compatibility via graph classes. In Michael A. Bekos and Michael Kaufmann, editors, *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 16–28. Springer, 2022. doi:10.1007/978-3-031-15914-5_2.

7 Selim G. Akl, Md. Kamrul Islam, and Henk Meijer. On planar path transformation. *Information Processing Letters*, 104(2):59–64, 2007. doi:10.1016/j.ipl.2007.05.009.

8 David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, 1996. doi:10.1016/0166-218X(95)00026-N.

9 Ahmad Biniaz, Anil Maheshwari, and Michiel Smid. Flip distance to some plane configurations. *Computational Geometry*, 81:12–21, 2019. doi:10.1016/j.comgeo.2019.01.008.

10 Carla Binucci, Fabrizio Montecchiani, Daniel Perz, and Alessandra Tappini. Flipping matchings is hard, 2025. URL: https://arxiv.org/abs/2503.02842.

11 Håvard Bakke Bjerkevik, Linda Kleist, Torsten Ueckerdt, and Birgit Vogtenhuber. Flipping non-crossing spanning trees. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2313–2325. SIAM, 2025. `doi:10.1137/1.9781611978322.77`.

12 Nicolas Bousquet, Lucas de Meyer, Théo Pierron, and Alexandra Wesolek. Reconfiguration of Plane Trees in Convex Geometric Graphs. In Wolfgang Mulzer and Jeff M. Phillips, editors, *40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:17, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.SoCG.2024.22`.

13 Nicolas Bousquet, Valentin Gledel, Jonathan Narboni, and Théo Pierron. A note on the flip distance between non-crossing spanning trees. *Computing in Geometry and Topology*, 2(1):8–1, 2023. `doi:10.57717/cgt.v2i1.36`.

14 Jou-Ming Chang and Ro-Yu Wu. On the diameter of geometric path graphs of points in convex position. *Information Processing Letters*, 109(8):409–413, 2009. `doi:10.1016/j.ipl.2008.12.017`.

15 Guilherme da Fonseca, Yan Gerard, and Bastien Rivier. On the longest flip sequence to untangle segments in the plane. In *WALCOM: Algorithms and Computation*, volume 13973, pages 102–112. Springer, 2023. `doi:10.1007/978-3-031-27051-2_10`.

16 M. R. Garey and David S. Johnson. The rectilinear steiner tree problem is NP-complete. *SIAM Journal of Applied Mathematics*, 32:826–834, 1977.

17 M. Carmen Hernando, Ferran Hurtado, Alberto Márquez, Merce Mora, and Marc Noy. Geometric tree graphs of points in convex position. *Discrete Applied Mathematics*, 93(1):51–66, 1999. `doi:10.1016/S0166-218X(99)00006-2`.

18 María del Carmen Hernando Martín, Fernando Alfredo Hurtado Díaz, and Marcos Noy Serrano. Graphs of non-crossing perfect matchings. *Graphs and combinatorics*, 18:517—-532, 2001. `doi:10.1007/s003730200038`.

19 Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, pages 333–346, 1999. `doi:10.1007/PL00009464`.

20 Iyad Kanj, Eric Sedgwick, and Ge Xia. Computing the flip distance between triangulations. *Discrete & Computational Geometry*, 58(2):313–344, 2017. `doi:10.1007/s00454-017-9867-x`.

21 Linda Kleist, Peter Kramer, and Christian Rieck. On the connectivity of the flip graph of plane spanning paths. In Daniel Kráľ and Martin Milanič, editors, *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 327–342. Springer, 2024. `doi:10.20382/jocg.v15i1a9`.

22 Charles L. Lawson. Transforming triangulations. *Discrete Mathematics*, 3(4):365–372, 1972. `doi:10.1016/0012-365X(72)90093-3`.

23 Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Computational Geometry*, 49:17–23, 2015. `doi:10.1016/j.comgeo.2014.11.001`.

24 Marcel Milich, Torsten Mütze, and Martin Pergel. On flips in planar matchings. *Discrete Applied Mathematics*, 289:427–445, 2021. `doi:10.1016/j.dam.2020.10.018`.

25 Torrie L Nichols, Alexander Pilz, Csaba D Tóth, and Ahad N Zehmakan. Transition operations over plane trees. *Discrete Mathematics*, 343(8):111929, 2020. `doi:10.1016/j.disc.2020.111929`.

26 Alexander Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 47(5):589–604, 2014. `doi:10.1016/j.comgeo.2014.01.001`.

27 Roberto Tamassia and Ioannis G. Tollis. A unified approach to visibility representation of planar graphs. *Discrete & Computational Geometry*, 1:321–341, 1986. doi:10.1007/BF02187705.

28 Uli Wagner and Emo Welzl. Connectivity of triangulation flip graphs in the plane. *Discrete & Computational Geometry*, 68(4):1227–1284, 2022. doi:10.1007/s00454-022-00436-2.

ParkView: Visualizing Monotone Interleavings

Thijs Beurskens[1], Steven van den Broek[1], Arjen Simons[1], Willem Sonke[1], Kevin Verbeek[1], Tim Ophelders[1,2], Michael Hoffmann[3], and Bettina Speckmann[1]

1 Dept. of Mathematics and Computer Science, TU Eindhoven, The Netherlands
[t.p.j.beurskens | s.w.v.d.broek | a.simons1 | w.m.sonke | k.a.b.verbeek | b.speckmann]@tue.nl

2 Dept. of Information and Computing Science, Utrecht University, The Netherlands
t.a.e.ophelders@uu.nl

3 Dept. of Computer Science, ETH Zürich, Switzerland
hoffmann@inf.ethz.ch

Abstract

We introduce ParkView: a schematic, scalable encoding for monotone interleavings on ordered merge trees. ParkView captures both maps of the interleaving using an optimal decomposition of the trees into paths. We prove several structural properties of monotone interleavings that enable a sparse visual encoding using a maximum of 6 colors for merge trees of arbitrary size.

Related Version arXiv:2501.10728

1 Introduction

A merge tree is a topological summary of a scalar field, which shows how the minima, maxima, and saddle points of the scalar field are connected (see Figure 1). The interleaving distance [4, 5, 7] is a similarity measure that captures how far two merge trees are from being isomorphic. Intuitively, it "weaves" the two trees together via two *shift maps* that take points from one tree to points a fixed distance higher in the other tree while preserving ancestry. Computing the interleaving distance is NP-hard [1] and in practice it is often desirable to introduce additional geometric constraints. The *monotone interleaving distance* [2] implements such constraints; it requires a prior ordering on the leaves of the merge trees that respects the tree structure. Given such an ordering, for example based on the spatial structure of the data, the monotone interleaving distance can be computed efficiently.

An *ordered merge tree* is a tree T equipped with a height function f and a total order on its leaves that respects T's structure. We think of T as a topological space; as such, we refer

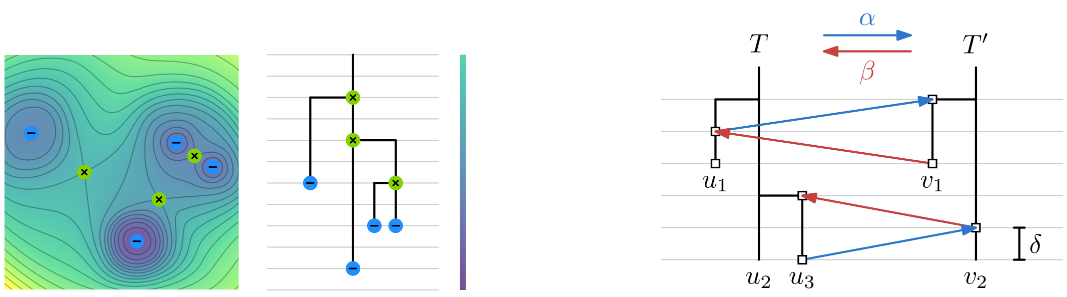


Figure 1 Left: a scalar field with its merge tree. Right: a δ-interleaving (α, β). We draw the trees rectilinearly; each horizontal line segment represents a single point, namely a non-leaf vertex.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

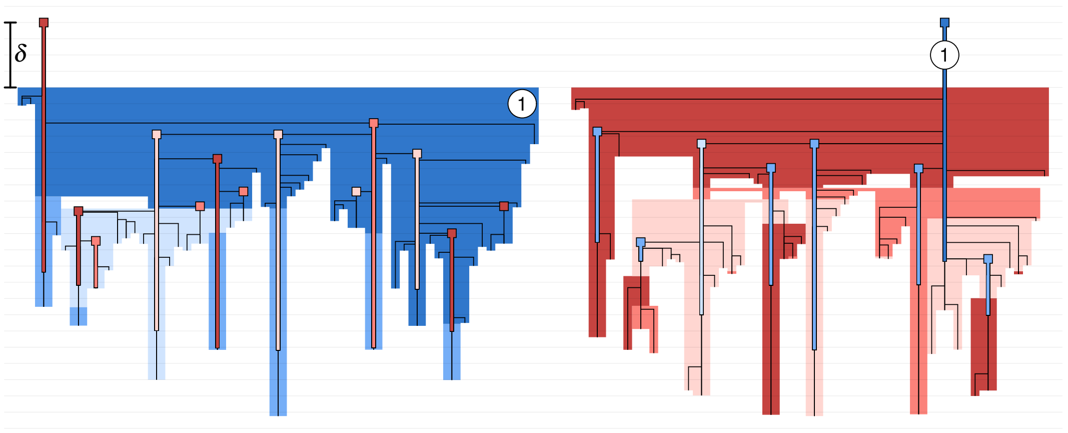


Figure 2 Example ParkView visualization of a monotone interleaving.

to not just the vertices, but also each point on the interior of an edge, as a *point* of T. The highest vertex of T is called the *root*, from which an edge extends upwards to infinity. The height function f has to be continuous and strictly increasing along each leaf-to-root path of T. A *monotone δ-shift map* α takes points in T and maps them continuously to points in T' exactly δ higher such that it preserves the order of any two points of T. A *monotone δ-interleaving* consists of two monotone δ-shift maps (α from T to T' and β from T' to T) such that for any point $x \in T$, the point $\beta(\alpha(x))$ is an ancestor of x and for any point $y \in T'$, the point $\alpha(\beta(y))$ is an ancestor of y. Figure 1 shows an example. The *monotone interleaving distance* is then the smallest δ for which a monotone δ-interleaving exists. In the remainder of this paper, we use "interleaving" to mean "monotone interleaving".

Interleavings on merge trees can have a complex structure, and hence to gain insight in their behavior, it is useful to visualize them. However, existing visualizations (e.g. [1, 3, 4, 5, 6, 7]) are mostly designed to visually explain the concept of interleavings on small examples, and not suitable for actual data exploration. We introduce ParkView: a schematic and scalable visual encoding for interleavings. To represent a shift map, ParkView decomposes the two merge trees into few components such that a component in one tree maps entirely to one component in the other tree. See Figure 2: the points in the left tree enclosed by shape 1 (a *hedge*) map to the points in the right tree on segment 1 (an *active path*). ParkView draws a merge tree rectilinearly, with the leaves drawn in separate columns according to the leaf order (Figure 3). The properties of a monotone interleaving allow us to match components

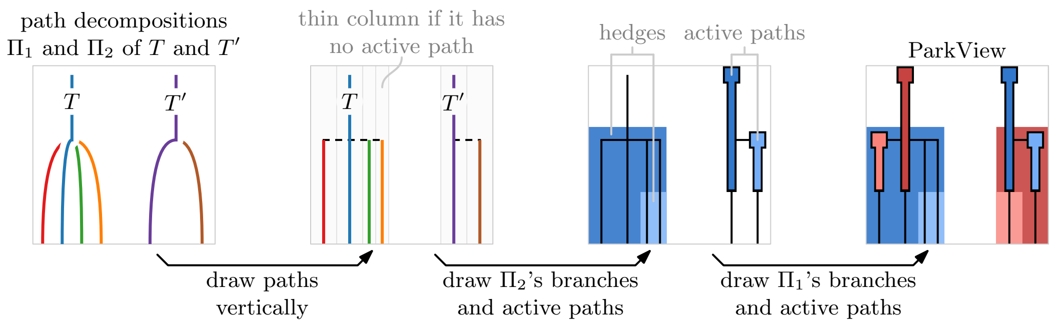


Figure 3 ParkView draws an interleaving (α, β) by superimposing drawings of heavy path-branch decomposition of both α and β.

left to right, based on the position of the lowest leaf for hedges and the x-position for active paths. Matching components are also assigned the same color. The drawings of the two shift maps combine and together show the interleaving.

In this paper, we detail two aspects of ParkView. First we define an optimal way of decomposing merge trees and show how to compute it (Section 2). Then we explain how we draw hedges and show that the set of hedges is 3-colorable (Section 3). The full version details the algorithmic pipeline for computing ParkView, and includes a showcase of ParkView on several real-world datasets.

2 Path-Branch Decomposition

The input for ParkView consists of two ordered merge trees T and T' and two shift maps α and β. We now describe the decomposition based on the shift map α; the decomposition based on β is symmetric. We decompose T' into a *path decomposition* Π: a set of height-monotone paths π that each start at a leaf (the *bottom* of π) and end at an internal vertex of T' (the *top* of π) or, for one path, at infinity. To make sure the paths of Π are disjoint and exactly cover T', we consider each path π to be open at its top. Alternatively, we can define a path decomposition bottom-up. For a vertex v of T', let the *up edge* be the one edge with increasing height incident to v, and let the *down edges* be the other edges incident to v. We now define a path decomposition by selecting, for each internal vertex v, one of the down edges of v as the *through edge* of v. The path decomposition is then built by starting a path at each leaf of T', and for each internal vertex v letting the incoming path from the through edge continue, while the incoming paths from the remaining down edges end at v.

Each path $\pi \in \Pi$ induces a *branch* B_π in T: the part of T that α maps to π. The branch B_π can either be empty, consist of a single connected component (a *simple branch*), or consist of multiple connected components (a *compound branch*) (see Figure 4). The complete set of branches B_π forms a decomposition of T, which we call the *branch decomposition* of T. Together, we call the paths in T' and the branches in T a *path-branch decomposition* for α. To minimize visual complexity, we now show how to construct an *optimal* path-branch decomposition: one that minimizes (1) the maximum number of branch components per path and (2) the total number of branch components.

As noted before, we can define a path decomposition of T' by selecting a through edge for each internal vertex v. For an edge e, let B_e be the part of T that α maps to the interior of e, and let the *weight* of e be the number of connected components of B_e. We define a *heavy path decomposition* by selecting the through edge of v to be a down edge of v with maximum weight. We now prove that a heavy path-branch decomposition is optimal. We refer to the highest edge π traverses as its *top edge*. We define the *size* of a branch B as the number of connected components it consists of. We first show that for a given path π, the size of its induced branch is equal to the weight of π's top edge.

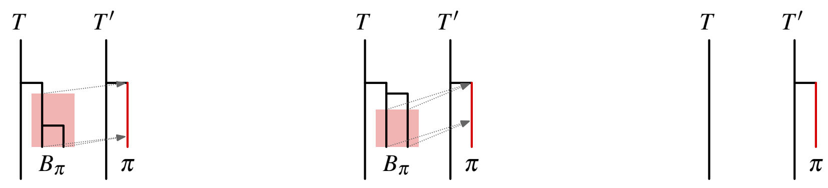


Figure 4 Examples of a simple branch, a compound branch, and an empty branch B_π.

▶ **Lemma 1.** *Let π be a path with top edge e. Then the size of B_π is equal to e's weight.*

Proof. Let v be the top of π and let $h := f(v) - \delta$. As e is in π, we have that $B_e \subseteq B_\pi$. It hence suffices to argue that each connected component C of B_π contains exactly one connected component of B_e. To show that C contains at least one connected component of B_e, we show that C contains a point x in B_e. Take any point $x' \in C$. If $\alpha(x')$ lies in the interior of e, then we take $x := x'$. Otherwise, we continuously follow the path from x' to the root of T. As α is continuous, the images of the points on the path (in T') also form a continuous path. Furthermore, as α is a δ-shift map, the images of these points also have a continuously increasing height value. It follows that there is a point x that maps to e. By definition $x \in B_e$ (and thus also in B_π). Furthermore, all points between x' and x on our path map to points on π in T'. Therefore, they are all part of B_π; hence, they are all part of the same connected component of B_π, namely C.

To show that C contains at most one connected component of B_e, assume for a contradiction that there are two distinct connected components C_1 and C_2 of B_e in C. As before, these components respectively contain points x_1 and x_2, both at height $h - \varepsilon$ for some $\varepsilon > 0$ chosen such that no vertices of T have height between h and $h - \varepsilon$. Now there is a path ρ from x_1 to x_2 entirely within C, as C is connected. There also is a distinct path ρ' from x_1 to x_2 via the lowest common ancestor x_3 in T of x_1 and x_2. Note that $f(x_3) \geq h$, so ρ' is not entirely within C; that is, $\rho \neq \rho'$. The union of ρ and ρ' hence contains a cycle, contradicting the fact that T is a tree. ◀

▶ **Theorem 2.** *Any heavy path-branch decomposition is optimal.*

Proof. Let Π be a path decomposition. Recall that Π selects one through edge for each vertex v in T'. Define the *cost* of v as the sum of the weights of v's down edges, excluding its through edge. As these edges are exactly the top edges ending at v, by Theorem 1, the cost of v is the number of branch components belonging to the paths ending at v. Then, the sum of costs of all vertices in T' is the total number of branch components induced by Π. This sum is minimized by minimizing the cost for each vertex v. This is achieved by maximizing the weight of its through edge, that is, picking a heavy edge as the through edge. A similar argument holds for minimizing the maximum number of branch components per path. ◀

3 Hedge Coloring

We represent each branch B_π by a *hedge* H_π: a rectilinear shape enclosing B_π (see Figure 5). Each hedge is a *histogram*: the union of a set of axis-aligned rectangles called *bars* whose tops are aligned. We call the height of the highest (lowest) point in a branch B_π its *top* (*bottom*) *height*. A hedge consists of three types of bars: *tree bars*, *fillers*, and *bridges*. For each path σ in the path decomposition of T that contains points in B_π, in the column of σ we add a *tree bar* whose bottom height is the height of the lowest point on σ that is in B_π.

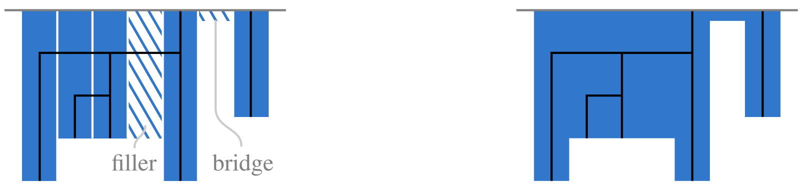
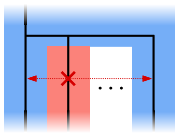


Figure 5 The types of bars that make up a hedge (left) and the resulting hedge (right).



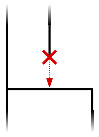


Figure 6 Illustrations of Observation 3 (left) and Observation 4 (right).

The union of these bars may not be connected; in this case, we connect consecutive leaves in the same branch component by adding *fillers* in the columns between them. The height of such a sequence of fillers is the smallest height of the two bars they connect (Figure 5). For a compound branch B_π, we draw its connected components like before, and then between them we add a *bridge*: a horizontal connector at the top of the hedge (Figure 5). The height of the bridge is less than the height of the shortest bar in the hedge.

A hedge H has a *left* (*right*) side which is the left (right) side of its leftmost (rightmost) bar. Two distinct hedges are *adjacent* if their boundaries, excluding corners, overlap. A hedge P is the *parent* of H if P is adjacent to the top of H; then H is a *child* of P.

It is desirable to use as few colors as possible for the hedges, while ensuring adjacent hedges have distinct colors. In fact, we show that the set of hedges in ParkView is 3-colorable. The proof makes use of three properties: hedges (i) are pairwise interior disjoint, (ii) have at most one parent, and (iii) have no hedge adjacent to the bottom of their longest bar. Our proofs of these properties rely on two observations about our drawing of T (see Figure 6).

▶ **Observation 3.** *No point of T is between two points of another branch at the same height.*

▶ **Observation 4.** *No leaves are positioned vertically above a horizontal segment.*

▶ **Lemma 5.** *Hedges in ParkView satisfy property (i).*

Proof sketch. Consider a horizontal line h that intersects a number of hedges. As hedges have a complicated shape, instead of studying the intersection of each hedge with h, we use Observation 3 to partition h into a number of interior disjoint intervals, one for each hedge. We then show that these intervals are supersets of the intersection of the corresponding hedge with h, from which it follows that the hedges are interior disjoint. ◀

▶ **Lemma 6.** *Hedges in ParkView satisfy property (ii).*

Proof sketch. For any hedge H_π, we can show that (a) it needs to have a point of T on the top, which is adjacent to some tree bar in a parent hedge, and (b) any other bars adjacent to the top of H_π need to be part of the same parent hedge. ◀

▶ **Lemma 7.** *Hedges in ParkView satisfy property (iii).*

Proof sketch. Let b be a longest bar in a hedge H_π. We can show that b is a tree bar: if it were a filler, this would violate Observation 4. We prove a key property: a tree bar that is a longest bar of its hedge has a leaf of T on its bottom. Hence, b has such a leaf. Now assume that there is another hedge H_ρ adjacent to the bottom of b. Then on the top of H_ρ, there is a point via which H_ρ connects to the rest of T. As each hedge has at most one parent (Lemma 6) this connection is via a bar b' of H_π. However, then b' is a longest tree bar. This contradicts our key property that b', being a longest tree bar, has a leaf on its bottom. ◀

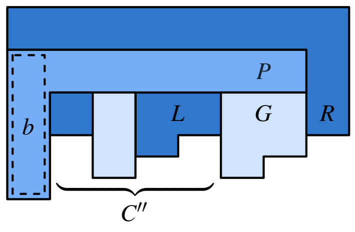


Figure 7 A set of histograms where P is the parent of G.

▶ **Theorem 8.** *Any set C of histograms that satisfies properties (i)–(iii) is 3-colorable.*

Proof. We use induction on $n = |C|$. The base case ($n = 1$) is trivial. Assume that C contains $n + 1$ histograms, and let G be a histogram whose top is lowest; it follows that no histogram in C is adjacent to the bottom side of any bar of G, and at most one histogram in C is adjacent to the left (or right) of G. Lastly, G can have at most one parent by (i), so G has at most three adjacent histograms.

The set $C' := C \setminus \{G\}$ still satisfies (i)–(iii) and has size n. By the induction hypothesis, C' is 3-colorable; fix a 3-coloring c_1 for C'. We edit c_1 into a 3-coloring for C. If the histograms adjacent to G use fewer than three colors, we use the third color for G to obtain a 3-coloring for C. Otherwise, let L and R be the histograms adjacent to the left and right of G, and let P be the parent of G. Since P, L, and R have distinct colors, we can assume without loss of generality that c_1 assigns colors 1, 2, and 3 to P, L, and R, respectively. By (iii) there is no histogram adjacent to the bottom of a longest bar of P, so P extends below the top of G. Without loss of generality, assume P extends left of G and call the rightmost such extending bar b (Figure 7). Consider the descendants C'' of P that lie to the left of G and to the right of b. As L is contained in C'', the set C'' is nonempty. This means that $C \setminus C''$ again satisfies (i)–(iii) and has size at most n, and is hence 3-colorable by the induction hypothesis. Let c_2 be a 3-coloring of $C \setminus C''$ such that without loss of generality P has color 1 and G has color 3. We now define a coloring c_3 for C where the histograms of $C \setminus C''$ take its color from c_2, and the histograms in C'' take their color from c_1.

Note that G and P are the only two histograms of $C \setminus C''$ that are adjacent to histograms in C''. So, one of four cases applies to any two adjacent histograms of C: (a) both lie in $C \setminus C''$, (b) both lie in C'', (c) one is P and the other lies in C'' or (d) one is G and the other lies in C'' (i.e., the other is L). For c_3 to be a 3-coloring, it suffices to show that in each case, c_3 assigns them distinct colors. In case (a), c_3 assigns the same colors as c_1. In case (b), c_3 assigns the same colors as c_2. In case (c), P has color 1 in both c_1 and c_2, so c_3 again assigns the same colors as c_1. In case (d), L has color 2 and G has color 3. ◀

Since hedges are histograms and satisfy (i)–(iii), the set of hedges in ParkView is 3-colorable.

Acknowledgments. Research on the topic of this paper was initiated at the 7th Workshop on Applied Geometric Algorithms (AGA 2023) in Otterlo, The Netherlands. Thijs Beurskens, Willem Sonke, Arjen Simons, and Tim Ophelders are supported by the Dutch Research Council (NWO) under project numbers OCENW.M20.089 (TB, WS), VI.Vidi.223.137 (AS), and VI.Veni.212.260 (TO).

References

1 P.K. Agarwal, K. Fox, A. Nath, A. Sidiropoulos, and Y. Wang. Computing the Gromov-Hausdorff distance for metric trees. *ACM Transactions on Algorithms*, 14(2):1–20, April 2018. doi:10.1145/3185466.

2 T. Beurskens, T. Ophelders, B. Speckmann, and K. Verbeek. Relating interleaving and Fréchet distances via ordered merge trees. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA25)*, pages 5027–5050. Society for Industrial and Applied Mathematics, 2025. `doi:10.1137/1.9781611978322.170`.

3 J. Curry, H. Hang, W. Mio, T. Needham, and O.B. Okutan. Decorated merge trees for persistent topology. *Journal of Applied and Computational Topology*, 6(3):371–428, February 2022. `doi:10.1007/s41468-022-00089-3`.

4 E. Gasparovic, E. Munch, S. Oudot, K. Turner, B. Wang, and Y. Wang. Intrinsic interleaving distance for merge trees. *La Matematica*, pages 1–26, 2024. `doi:10.1007/s44007-024-00143-9`.

5 D. Morozov, K. Beketayev, and G. Weber. Interleaving distance between merge trees. Manuscript (accessed on 06-03-2025), 2013. URL: `https://mrzv.org/publications/interleaving-distance-merge-trees/manuscript/`.

6 M. Pegoraro. A graph-matching formulation of the interleaving distance between merge trees. `arXiv:2111.15531`.

7 E.F. Touli and Y. Wang. FPT-algorithms for computing the Gromov-Hausdorff and interleaving distances between trees. *Journal of Computational Geometry*, 13(1):89–124, April 2022. `doi:10.20382/jocg.v13i1a4`.

A collapse algorithm for farthest Voronoi diagrams in three dimensions*

Evanthia Papadopoulou, Martin Suderland, and Zeyu Wang

Faculty of Informatics, Università della Svizzera italiana (USI), Switzerland
`{evanthia.papadopoulou, martin.suderland, zeyu.wang}@usi.ch`

Abstract

We devise a *collapse algorithm*, which computes the farthest Voronoi diagram of convex sites induced by a convex distance function in 3D. It is a useful tool in clarifying the structure of this diagram, and can express its combinatorial complexity in terms of the algorithm's number of events. Moreover, we show that the Euclidean distance function along a trisector of lines in 3D has at most 4 local maxima and 8 local minima. This answers the question on how to find the smallest sphere touching three lines. We show that this sphere touches the lines along a great circle.

1 Introduction

Voronoi diagrams are among the most fundamental space partitioning structures in Computational Geometry. Given a set S of n objects in some space, called sites, the *nearest* (respectively, *farthest*) Voronoi diagram of S decomposes the underlying space into regions, that have the same closest (resp., farthest) site. In this note, we describe a very general algorithm for the construction of farthest Voronoi diagrams in $\mathbb{R}^3$.

Voronoi diagrams in the Euclidean plane have been intensively studied [5, 8, 9, 11, 13, 14, 17], in three dimensions, however, Voronoi diagrams are far less understood, especially if sites are non-point objects. A general framework for studying Voronoi diagrams in $\mathbb{R}^d$ is through the *arrangements* of the distance functions of the given sites in $\mathbb{R}^{d+1}$ [11]. The lower (resp., upper) envelope of these distance functions, projected back to $\mathbb{R}^d$, yields the nearest (resp., farthest) Voronoi diagram of the given sites. If the distance functions are simple-enough hypersurfaces, then the complexity of either Voronoi diagram is $O(n^{d+\epsilon})$, for any $\varepsilon > 0$ [18]. For $d = 3$, the diagram can also be computed within the same bound [1]. If sites are $(d-2)$-flats, a lower bound of $\Omega(n^{d-1})$ has been given by Aronov [2]. Tighter combinatorial bounds in $\mathbb{R}^3$ are known for restricted cases [4, 10, 15, 16]. These include $O(c^3 n^{2+\varepsilon})$ for lines with a constant number of c orientations [16] and $O(n^{2+\varepsilon})$ for parallel halflines [4]. The unbounded features of the order-k (and farthest) Voronoi diagrams of lines and line segments in $\mathbb{R}^d$ have been studied by Barequet et al. [7]. They are encoded in the Gaussian map, a map on the sphere of directions, which has complexity $O(\min\{k, n-k\}n^{d-1})$. The complexity of the unbounded features of the farthest Voronoi diagram of lines in $\mathbb{R}^d$ is $\Theta(n^{d-1})$.

Tighter bounds are known for convex distance functions induced by a polyhedron of constant complexity. The size of the nearest Voronoi diagram is $O(n^2\alpha(n)\log(n))$ and a lower bound of $\Omega(n^2\alpha(n))$ can be realized [10], when the sites are lines. For disjoint line segments or polyhedra with n vertices, the complexity is $O(n^2\alpha(n)\log n)$ and $O(n^{2+\varepsilon})$ respectively [15]. For convex polyhedral sites the farthest Voronoi diagram has up to $\Theta(n^{\lceil \frac{d}{2} \rceil})$ complexity and can be computed in time proportional to the bound plus an $O(n\log n)$ term [6]. The diagram has constant complexity for point sites or sites with a constant number of orientations.

* This research was supported by the Swiss National Science Foundation (SNF), project 200021E_201356. Martin Suderland was also supported by SNF project P500PT_206736/1.

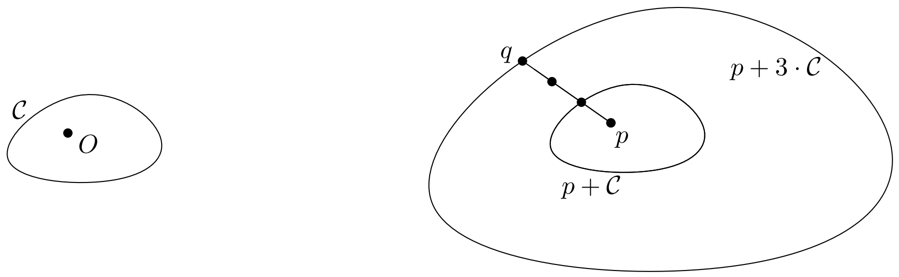


Figure 1 Convex distance induced by the convex set $\mathcal{C}$ containing the origin O: $d(p, q) = 3$

In this abstract, we devise a *collapse algorithm*, which computes the farthest Voronoi diagram of convex sites induced by a convex distance function in 3D. It is a useful tool in clarifying the structure of this diagram, and can express its combinatorial complexity in terms of the algorithm's number of events. The algorithm discovers features of the diagram in decreasing distance to their farthest site. This motivates to look at how this distance evolves along the edges of the diagram. We show that the Euclidean distance function along a trisector of lines in 3D has at most 4 local maxima and 8 local minima. This answers the question on how to find the smallest sphere touching three lines, which is of independent interest. We show that this sphere touches the lines along a great circle.

2 Preliminaries

Let $S = \{s_1, ..., s_n\}$ be a set of n disjoint convex sites in $\mathbb{R}^3$. Any bounded convex set $\mathcal{C} \subset \mathbb{R}^3$ which contains the origin in its interior can be used to define a so-called *convex distance*, between two points $p, q \in \mathbb{R}^3$: $d(p, q) = \inf_{t \geq 0} \{ t \mid q \in p + t \cdot \mathcal{C} \}$. In other words, $d(p, q)$ describes the amount $t \geq 0$ by which $\mathcal{C}$, when being placed at p, has to be scaled so as to cover q; see Fig. 1. This is a directed distance, i.e. when $\mathcal{C}$ is not point-symmetric to the origin, then $d(p, q) \neq d(q, p)$ for some pair of points $p, q \in \mathbb{R}^3$. The distance $d(p, s)$ from a point $p \in \mathbb{R}^3$ to a site $s \in S$ is defined as $d(p, s) = \min\{d(p, q) \mid q \in s\}$.

The ***farthest Voronoi region*** of $s \in S$ is the set of points in $\mathbb{R}^3$ whose distance to s is larger than to any other site in S. It is denoted as $\text{freg}(s, S) = \{p \in \mathbb{R}^3 \mid \forall s' \in S : d(p, s) \geq d(p, s')\}$. The farthest regions of S induce a subdivision in $\mathbb{R}^3$; the induced cell complex is called the ***farthest Voronoi diagram*** of S, denoted by $\text{FVD}(S)$. It consists of vertices, edges, faces, and 3-dimensional cells; we refer to the latter simply as cells. The complexity of a cell complex M is the total number of its features, denoted as $|M|$. The i-sector of i sites in $\mathbb{R}^3$ is the locus of points at equal distance to the i sites. Two trisectors (3-sectors) are called *related* if they are defined by exactly four distinct sites.

We say two cell complexes M and M' have the same ***topology*** if there is an isomorphism that maps features in M to features of the same dimension in M' and maintains all incidence relations, i.e., two features are incident in M if and only if their images are incident in M'.

▶ **Definition 2.1.** For a finite cell complex M, let B be a topological ball large enough to intersect any cell of M of any dimension in one connected component. Let Γ denote the boundary of B. The intersection $M \cap \Gamma$ is called the Γ***-map*** of M, denoted by $\Gamma\text{M}(M)$.

The topology of $\Gamma\text{M}(M)$ is invariant, independently of the specific choice of Γ, see Fig. 2. When considering the $\text{FVD}(S)$ of a set of lines as sites with the Euclidean distance, the Γ-map can be obtained from the Gaussian map [7] by doing some small modifications.

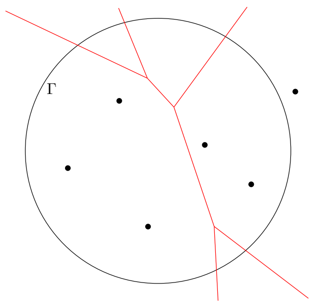


Figure 2 The Euclidean farthest Voronoi diagram (in red) of a set of points S in the plane together with a possible choice of Γ for the definition of $\Gamma\mathrm{M}(\mathrm{FVD}(S))$.

We assume the following general position: (1) any k-sector has dimension $4-k$ for all $1 \leq k$; (2) no circle is touching 4 sites; (3) the number of local minima and maxima along a trisector of any 3 sites is bounded by a constant. The general position assumptions ensure the following properties: by assumption (1), vertices (resp. edges, faces, cells) of the FVD are equidistant to exactly 4 (resp. 3,2,1) sites; by (2), related trisectors intersect transversely, which is shown in the sequel; by (3), the analysis of the algorithm in Section 4 simplifies.

▶ **Lemma 2.2.** *If two related trisectors are intersecting tangentially, then there exists a circle which is touching 4 sites.*

3 On the Euclidean distance function along a trisector

In this section we consider the Euclidean distance. It was shown that the trisector of lines is a quartic consisting of four unbounded branches [12, 16], assuming that the 3 lines are not parallel to the same plane, and pairwise skew. In general, edges of a Voronoi diagram in $\mathbb{R}^d$ are part of a $(d-1)$-sector of the sites. We are interested in the distance function, which maps any point p along this $(d-1)$-sector to the distance between p and the involved sites. In the plane, the distance function along the bisector of line segments has exactly one local minimum but no local maximum. It is surprising that in 3D the distance function along a trisector of lines can have local maxima, see Figs. 3 and 4. Phrased differently, branches of the trisector can have more than one local minimum.

▶ **Theorem 3.1.** *The Euclidean distance function along the trisector of lines in $\mathbb{R}^3$ can admit at most 4 local maxima and 8 local minima. These bounds are tight.*

Imagine sliding the center of a sphere along the trisector, such that it touches the defining sites. A local maximum corresponds to a point at which the radius of the sphere is decreasing in both directions along the trisector. If one wants to find the smallest sphere, which is simultaneously touching 3 lines, one has to compute the possibly 8 local minima of the distance function and pick the smallest. There is a nice geometric interpretation of these local extrema.

▶ **Theorem 3.2.** *An extremum of the Euclidean distance function along the trisector of convex sites in $\mathbb{R}^3$ corresponds to the center of a sphere $\mathbb{S}$, which touches the 3 sites along a great circle of $\mathbb{S}$.*

Another surprising fact is that the trisector of sites can be bounded if considering line segments as sites, see Fig. 5.

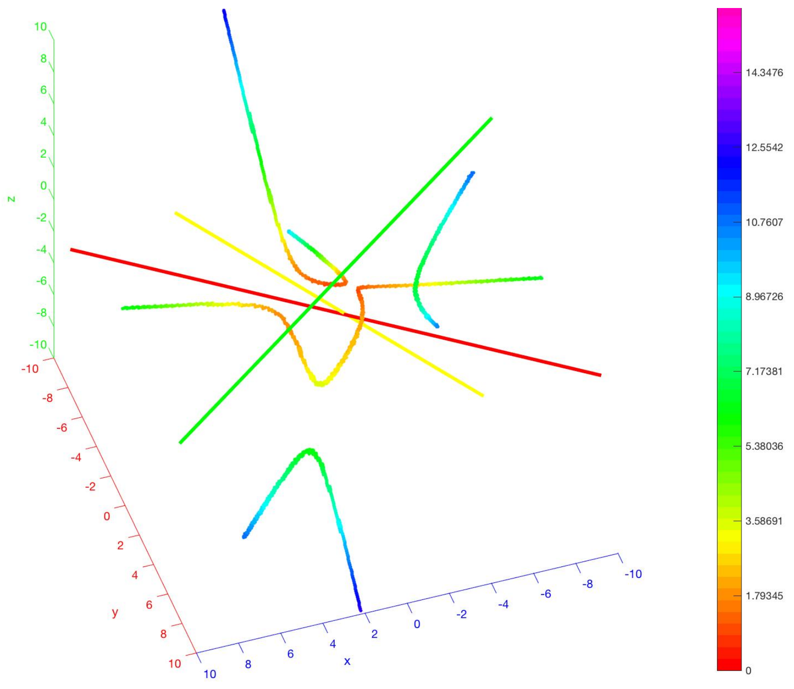


Figure 3 The trisector of three lines (red, green, yellow). The colors along the trisector encode the distance to the lines. In this example, the distance function along the middle branch admits a local maximum, as the color coding changes from orange to yellow to orange. A rotating animation of the trisectors shown in Figs. 3 and 5 can be found on https://compgeom.inf.usi.ch/research.html.

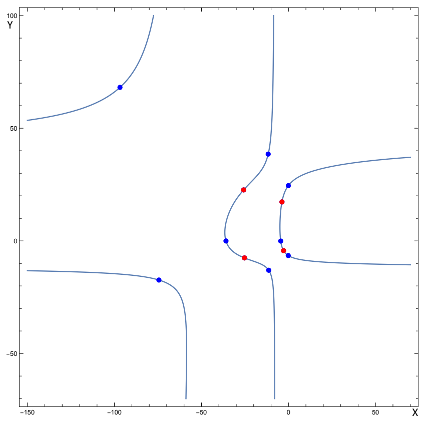


Figure 4 A projected trisector with highlighted local **minima** in blue and local **maxima** in red.

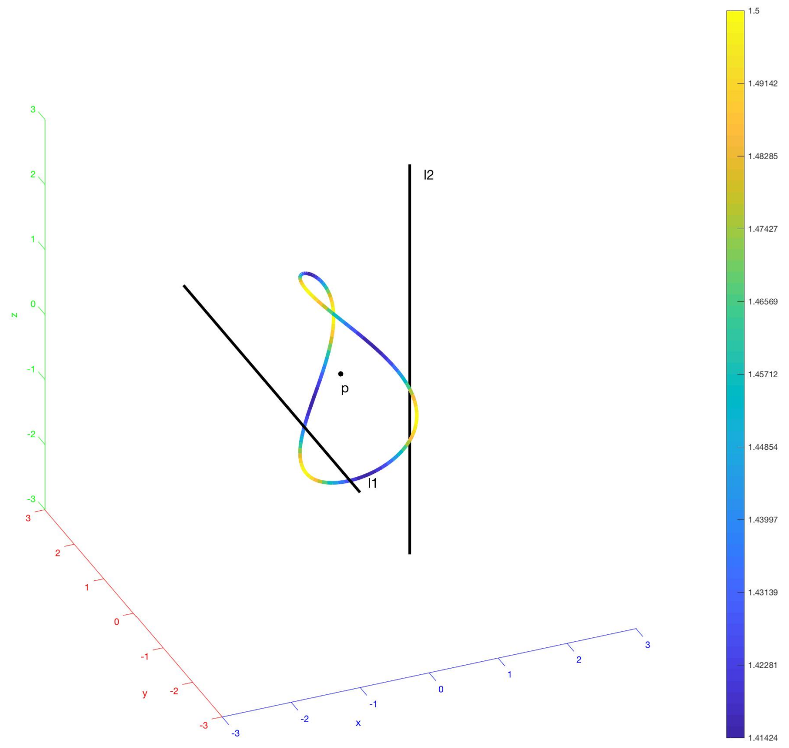


Figure 5 The trisector of 2 lines $l_1 : x = -1 \wedge z = 0$ and $l_2 : x = 1 \wedge y = 0$ and point $p = (0, 0, 0)$.

4 A collapse algorithm for the farthest Voronoi diagram

A *collapse* algorithm to compute the Euclidean farthest Voronoi diagram of line segments in the plane has been described by Aurenhammer et al. [3]. In this abstract, we give a generalization to convex sites $S \subset \mathbb{R}^3$ with any convex distance. The algorithm proceeds in two steps. In the first phase, the unbounded features of the FVD are computed, which appear in the Γ-map. In the second phase, the algorithm discovers the features of the FVD in decreasing distance from their farthest site. For such a feature of the FVD, the distance to its farthest site is called the *priority* of this feature. The algorithm finds the features of the diagram in decreasing priority. We maintain a map of all points that have the same priority:

▶ **Definition 4.1** (Shrinking map). For a set of sites S and some $\lambda \in \mathbb{R}$, let the *shrinking map* M_λ be the intersection of the hyperplane $x_4 = \lambda$ and the upper envelope of the distance functions $\{d(.,s) : s \in S\}$.

Due to the convexity of the distance functions $\{d(.,s) : s \in S\}$, the upper envelope of these functions is also a convex function. This implies that the shrinking map M_λ is a topological sphere, because it is the intersection of a convex function $\mathbb{R}^3 \to \mathbb{R}$ with a horizontal hyperplane $x_4 = \lambda$. The algorithm scans the upper envelope as it sweeps a hyperplane, starting at $x_4 = \infty$ and moving down, until the entire envelope is scanned. Throughout the algorithm, we maintain a topologically correct representation M of the shrinking map M_λ. A change in the topology of map M corresponds to an event of the algorithm. The list of possible events is given in Fig. 6. There are five types of events, in contrast to only one type in 2D. Two of these events involve a vertex of the FVD, which we describe next.

Deletion event. 3 faces (blue, green, purple) on M are surrounding a forth one (red) just before the event. The red face shrinks until it completely disappears at the event. The event corresponds to a vertex in the FVD. After the event, the red face together with its bounding vertices and edges is removed from M and replaced by one vertex.

Swap event. Before the event, the map M locally shows 4 faces (blue, red, green, purple in cyclic order) of which the opposite faces blue and green are sharing a shrinking edge. At the event, all 4 faces are touching at a common point, which corresponds to a vertex in the FVD. After the event, the faces red and purple are sharing a growing edge instead.

Local minimum event. A shrinking face (red) is fully bounded by two other faces (blue, green) of M. At the event, the red face disappears, which corresponds to a local minimum of the trisector between the red, blue and green sites in the FVD. After the event only an edge separating the blue and green face remains locally.

Local maximum event. Before the event, two faces (blue, green) are locally separated by another face (red) on M. At the event, the red face is getting split into 2 faces and all four faces share a common point. In the FVD, this point corresponds to a local maximum of the trisector of the 3 involved sites. After the event, the map has 4 faces (blue, red, green, red in cyclic order) and the faces blue and green share an edge.

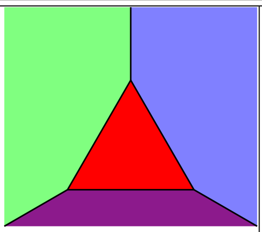
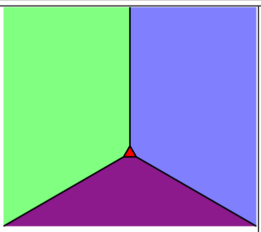
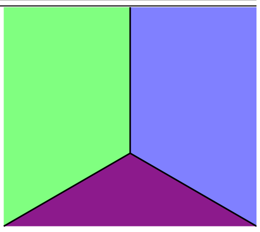
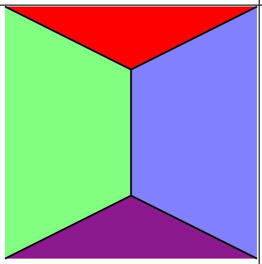
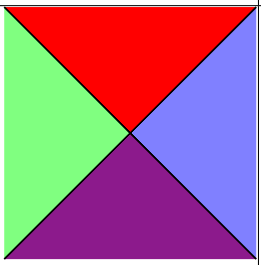
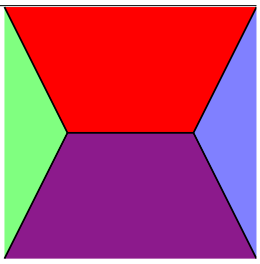
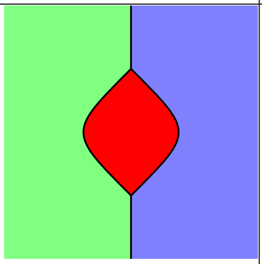
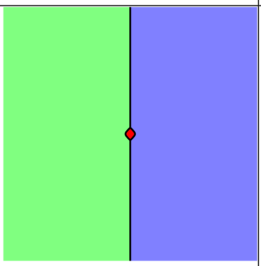
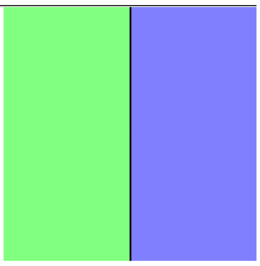
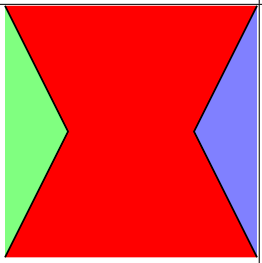
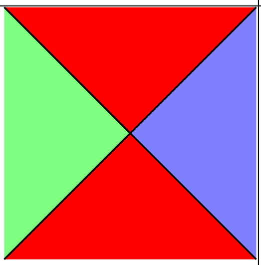
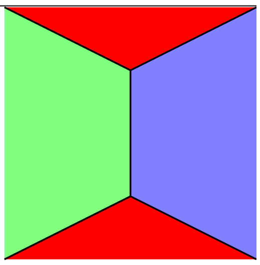
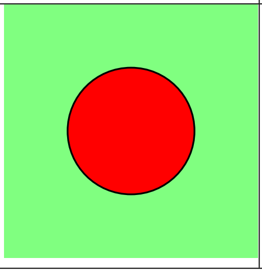
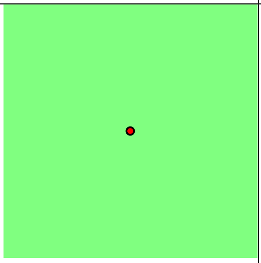
Event name	before	at	after
Deletion			
Swap			
Local minimum			
Local maximum			
Stop event			end

Figure 6 List of collapse events. The columns 2-4 show how the shrinking map changes locally at each event, where a piece of the shrinking map is depicted before, at, and after the event happens.

Stop event. One face (red) is fully enclosed by another face (green) on M. The red face is shrinking until it disappears at the event. This event corresponds to the minimum of the priority function on the red-green bisector in the FVD. We will prove that if this event occurs, then it is the last one.

Apart from the stop event, other events can also be the last ones to be processed. That is the local minimum (resp. deletion) event, in case that the global minimum of the priority function is realized at an edge (resp. vertex) of the FVD. Note that the global minimum of the convex upper envelope might not be unique, but that there can be an entire connected set of points which are global minima. In this case, the last step of the algorithm finishes constructing the diagram, by adding a piece of an edge or face.

▶ **Theorem 4.2.** *The list of events in Fig. 6 is complete.*

Next, we discuss how to express the combinatorial complexity of the FVD in terms of the number of events, which happen during the collapse algorithm. The Γ-map is a planar graph, on which Euler's formula holds, thus, the number of faces, edges, and vertices on the Γ-map are linearly related.

Let n_d, n_s, n_m, n_M be the number of times the **d**eletion, **s**wap, local **m**inimum, and local **M**aximum event happens during the collapse algorithm. The complexity of the entire farthest Voronoi diagram is asymptotically just the total number of events happening during the collapse algorithm, i.e. $|\mathrm{FVD}| = \Theta(n_d + n_s + n_m + n_M)$. Even though an edge may contain many local minima and maxima, it is only counted at most a constant number of times due to our general position assumption.

One can observe that only the deletion, local minimum and stop events delete faces from the shrinking map. Each deletion and local minimum event removes exactly one face from the shrinking map apart from the last event of the collapse algorithm, which removes between 2 and 4 faces. On the other hand, the only event, that can create an extra face on the shrinking map is the local maximum event. Each face starting on the ΓM(FVD) or getting later created at a local maximum event, needs to be removed before the collapse algorithm finishes. Thus, based on counting the number of faces on the shrinking map, we derive the condition $|\Gamma\mathrm{M}(\mathrm{FVD})| + n_M - n_d - n_m \in \{2, 3, 4\}$.

▶ **Theorem 4.3.** *For a set of convex sites S, we have $|\mathrm{FVD}| = \Theta(|\Gamma\mathrm{M}(\mathrm{FVD})| + n_s + n_M)$.*

For n lines as sites with the Euclidean distance, $|\Gamma\mathrm{M}(\mathrm{FVD})| = \Theta(n^2)$ follows from [7].

References

1 Pankaj K Agarwal, Boris Aronov, and Micha Sharir. Computing envelopes in four dimensions with applications. In *Proceedings of the tenth annual symposium on Computational geometry*, pages 348–358, 1994.

2 Boris Aronov. A lower bound on Voronoi diagram complexity. *Information Processing Letters*, 83(4):183–185, 2002.

3 Franz Aurenhammer, Robert L. S. Drysdale, and Hannes Krasser. Farthest line segment Voronoi diagrams. *Information Processing Letters*, 100(6):220–225, 2006.

4 Franz Aurenhammer, Bert Jüttler, and Günter Paulini. Voronoi diagrams for parallel halflines and line segments in space. In *28th International Symposium on Algorithms and Computation (ISAAC 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

5 Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi diagrams and Delaunay triangulations.* World Scientific Publishing Company, 2013.

6 Franz Aurenhammer, Evanthia Papadopoulou, and Martin Suderland. Piecewise-linear farthest-site Voronoi diagrams. In *32nd International Symposium on Algorithms and Computation (ISAAC 2021)*, pages 327–345. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

7 Gill Barequet, Evanthia Papadopoulou, and Martin Suderland. Unbounded regions of high-order Voronoi diagrams of lines and line segments in higher dimensions. *Discrete & Computational Geometry*, pages 1–29, 2023.

8 Jean-Daniel Boissonnat, Micha Sharir, Boaz Tagansky, and Mariette Yvinec. Voronoi diagrams in higher dimensions under certain polyhedral distance functions. In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 79–88, 1995.

9 Bernard Chazelle. An optimal convex hull algorithm and new results on cuttings (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico*, pages 29–38. IEEE Computer Society, 1991.

10 L Paul Chew, Klara Kedem, Micha Sharir, Boaz Tagansky, and Emo Welzl. Voronoi diagrams of lines in 3-space under polyhedral convex distance functions. *Journal of Algorithms*, 29(2):238–255, 1998.

11 Herbert Edelsbrunner and Raimund Seidel. Voronoi diagrams and arrangements. *Discrete & Computational Geometry*, 1:25–44, 1986.

12 Hazel Everett, Daniel Lazard, Sylvain Lazard, and Mohab Safey El Din. The Voronoi diagram of three lines. *Discrete & Computational Geometry*, 42(1):94–130, 2009.

13 Christian Icking and Lihong Ha. A tight bound for the complexity of Voroni diagrams under polyhedral convex distance functions in 3d. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 316–321, 2001.

14 Victor Klee. On the complexity of d-dimensional Voronoi diagrams. *Archiv der Mathematik*, 34(1):75–80, 1980.

15 Vladlen Koltun and Micha Sharir. Polyhedral Voronoi diagrams of polyhedra in three dimensions. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 227–236, 2002.

16 Vladlen Koltun and Micha Sharir. Three dimensional Euclidean Voronoi diagrams of lines with a fixed number of orientations. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 217–226, 2002.

17 Raimund Seidel. On the number of faces in higher-dimensional Voronoi diagrams. In *Proceedings of the third annual symposium on Computational geometry*, pages 181–185, 1987.

18 Micha Sharir. Almost tight upper bounds for lower envelopes in higher dimensions. *Discrete & Computational Geometry*, 12:327–345, 1994.

Containment results on points and spheres*

Andrea de las Heras-Parrilla[1], David Flores-Peñaloza[2], Clemens Huemer[1], and David Orden[3]

1 **Universitat Politècnica de Catalunya**
`andrea.de.las.heras@upc.edu`, `clemens.huemer@upc.edu`

2 **Facultad de Ciencias, Universidad Nacional Autónoma de México**
`dflorespenaloza@ciencias.unam.mx`

3 **Universidad de Alcalá**
`david.orden@uah.es`

Abstract

Let S be a set of n points in general position in $\mathbb{R}^d$. We show several containment results for points from S in spheres determined by $d+1$ points of S. Among them, we prove a Delaunay-type criterion for point sets in $\mathbb{R}^3$. Also, we show bounds on the expected number of points from S contained in a sphere determined by four points chosen uniformly at random from $S \subset \mathbb{R}^3$. A tight upper bound construction is provided, obtained by inversion of points on the moment curve. We also show a lower bound and prove that it is best possible for $n \leq 7$. In order to do so, we solve the recurrence relation $T(n) = \left\lceil \frac{n}{n-5} T(n-1) \right\rceil$ with base case $T(7) = 29$. This is of independent interest, since most recurrence relations of this type seem not to have a solution in closed form.

1 Introduction

Let S be a set of $n \geq d+2$ points in general position in $\mathbb{R}^d$, $d \geq 2$, meaning no m of them lie on a $(m-2)$-dimensional flat for $m = 2, 3, \ldots, d+1$ and no $d+2$ of them lie on the same $(d-1)$-sphere. We show several containment results for points from S in the open balls having as boundary spheres determined by $d+1$ points from S. With a slight abuse of notation, usual in the literature, in the following we will say *sphere* instead of *open ball* for this containment relationship.

First, we prove a Delaunay-type criterion for point sets in $\mathbb{R}^3$. The well-known *empty circle property* of the Delaunay triangulation in $\mathbb{R}^2$, see e.g. Lemma 9.4 in [4], states that given a set $S = \{a, b, c, d\}$ of four points in convex position in the plane, then exactly two of the four circles passing through three points of S contain the fourth point of S; and if the line passing through two points a and b of S separates c and d, then the circle through a, b, c contains d if and only if the circle through a, b, d contains c. This criterion is commonly used to characterize the Delaunay triangulation of a set S of n points in $\mathbb{R}^2$ as the set of triangles with vertices from S, whose circumcircles are empty of other points from S. Delaunay in his paper [7] from 1934 stated this more generally for $\mathbb{R}^n$, $n \geq 2$. We obtain a statement similar to the Delaunay criterion, for five points in $\mathbb{R}^3$, given in Section 2.

Second, we study the expected number $\mathbb{E}(X_{S,d})$ of points from S that are contained in the sphere passing through $d+1$ different points from S, chosen uniformly at random. $\mathbb{E}(X_{S,d})$ is determined by the vector $(s_0, s_1, \ldots, s_{n-d-1})$, where s_k is the number of spheres passing

* Andrea de las Heras-Parrilla, Clemens Huemer and David Orden were supported by Grant PID2023-150725NB-I00 funded by MICIU/AEI/10.13039/501100011033. David Flores-Peñaloza was supported by Grant PAPIIT-IN115923, UNAM, México. Clemens Huemer was also supported by project Gen. Cat. DGR 2021-SGR-00266. The authors acknowledge valuable comments and references from the anonymous reviewers.

through $d+1$ points of S that enclose exactly k other points from S, for $k = 0, \dots n-d-1$. Clearly, $\sum_{k=0}^{n-d-1} s_k = \binom{n}{d+1}$. Then,

$$\mathbb{E}(X_{S,d}) = \frac{\sum_{k=0}^{n-d-1} k \cdot s_k}{\binom{n}{d+1}}. \tag{1}$$

The expression $\sum_{k=0}^{n-d-1} k \cdot s_k$ can also be interpreted as the number of (p, Q) pairs, such that p is a point of S, and Q is a sphere induced by $d+1$ points of $S \backslash \{p\}$ containing p in its interior. In dimension $d = 2$, $\mathbb{E}(X_{S,2})$ is equivalent to the *rectilinear crossing number* [29] of S, denoted $\overline{cr}(S)$, via the known relation, first obtained by Urrutia [31], also see [12]:

$$\sum_{k=0}^{n-3} k \cdot s_k = \binom{n}{4} + \overline{cr}(S). \tag{2}$$

In this paper we mainly focus on dimension $d = 3$. We define $S_n = \min \sum_{k=0}^{n-4} k \cdot s_k$, where the minimum is taken over all sets S of n points in general position in $\mathbb{R}^3$.

In Section 3 we show a lower bound of $S_n \geq 2 \left\lfloor \frac{\binom{n}{5}}{5} \right\rfloor + \binom{n}{5} - 2 \left\lfloor \frac{n}{25} \right\rfloor$ for each $n \geq 5$. We have found point sets showing that this bound is best possible for $n \leq 7$; i.e., $S_5 = 1$, $S_6 = 8$, and $S_7 = 29$. Other found point sets show that $S_8 \leq 80, S_9 \leq 189,$ and $S_{10} \leq 376$.

To prove the bound on S_n, we present a solution in closed form of the recurrence relation

$$T(n) = \left\lceil \frac{n}{n-a} T(n-1) \right\rceil \qquad \text{for } n > b, \text{ and } T(b) = c,$$

with $a = 5$, $b = 7$, $c = 29$. Interestingly enough, Conway et al. [6] needed to solve, for a different problem, the case with $a = 3$, $b = 4$, $c = 1$, and stated that most recurrence relations of this shape, for given integers a, b, c, seem not to have a solution in closed form, leaving as an open problem to characterize those which do.

We will consider all five-tuples of points from S. For a set $\mathcal{S}$ of five points in $\mathbb{R}^3$ we say that $\mathcal{S}$ is of Type A if $s_0 = 4$ (then $s_1 = 1$), of Type B if $s_0 = 3$ (then $s_1 = 2$), and of Type C if $s_0 = 2$ (then $s_1 = 3$). Calling A, B, C the number of five-tuples of each type we can write

$$\sum_{k=0}^{n-4} k \cdot s_k = 1 \cdot A + 2 \cdot B + 3 \cdot C. \tag{3}$$

Note that these are the only possible types, because each sphere counted by s_0 corresponds to a simplex of the Delaunay triangulation of $\mathcal{S}$ and the number of simplices in a triangulation of n points with h of them on the boundary of the convex hull is between $n-3$ and $\binom{n-1}{2} - h + 2$ [11]. In particular, a set of five points in non-convex position is of Type A, whereas there are two types, B and C, of sets of five points in convex position.

In Section 4 we show that for a set S of n points on the moment curve in $\mathbb{R}^3$, all its five-tuples of points are of Type B. Then, among all sets S of n points in convex position, points on the moment curve minimize $\sum_{k=0}^{n-4} k \cdot s_k$. Let us also remark that the order type [13] of a point set does not determine the types A, B, C of all of its five-tuples of points. In particular, there are cyclic polytopes, i.e., point sets that have the same order type as a set of points on the moment curve, not all whose five-tuples are of Type B. We also prove that there exist sets of n points all of whose five-tuples are of Type C and thus, by (3), maximize $\sum_{k=0}^{n-4} k \cdot s_k$ among all sets of n points in general position in $\mathbb{R}^3$. Interestingly, these point sets are obtained by applying inversion to the points on an arc of the moment curve.

Finally, in Section 5 we consider sets S of points in $\mathbb{R}^d$, $d \geq 2$. Let $P_d(S)$ be the probability that the sphere passing through $d+1$ points chosen uniformly at random from S, contains another point chosen uniformly at random from the remaining points of S. We define $P_d(n)$ as the minimum of $P_d(S)$ among all sets S of n points in general position in $\mathbb{R}^d$, and $P_d^* = \lim_{n\to\infty} P_d(n)$. We prove that this limit exists for each fixed dimension d. For $d = 2$, we observe that P_2^* is equivalent to the *rectilinear crossing number constant* $\overline{\nu}^*$, see e.g. [30], namely $P_2^* = \frac{1+\overline{\nu}^*}{4}$ by using Equation (2). Then it is not surprising that the proof for existence of $\overline{\nu}^*$ from [30] extends smoothly to a proof for existence of P_d^* for $d > 2$. For dimension $d = 3$, we show the lower bound $P_3^* \geq \frac{7}{25}$. Other research on containment results on points and spheres was mainly carried out in dimension $d = 2$, see e.g. [2, 5, 10, 15, 16, 21, 25], with others, but noticeably fewer, for $d \geq 3$, see e.g. [3, 8, 9, 24, 27]. Due to lack of space, most proofs are omitted.

2 A Delaunay-type criterion in $\mathbb{R}^3$

We say that a plane π separates two points d and e, if d and e do not lie in the same (closed) half-space bounded by π. We say that a triangle $\Delta(a, b, c)$ with vertices a, b, and c separates two points d and e, if the plane π passing through a, b, and c, separates d and e. We denote the sphere passing through four points a, b, c, and d with $\bigcirc(a, b, c, d)$.

▶ **Lemma 2.1.** *Let $S = \{a, b, c, d, e\}$ be a set of five points in general and convex position in $\mathbb{R}^3$, such that the plane π passing through a, b, c separates d and e. Then the sphere $\bigcirc(a, b, c, d)$ contains e in its interior if, and only if, the sphere $\bigcirc(a, b, c, e)$ contains d in its interior.*

Proof. We will use determinant tests, see e.g. [1, 7, 14, 26], or [23], Equation (4.7.1), for a detailed discussion. Denote a point $p \in \mathbb{R}^3$ as $p = (x_p, y_p, z_p)$. It is well known that a point e lies on the sphere $\bigcirc(a, b, c, d)$ passing through four other points $a, b, c, d \in \mathbb{R}^3$ if, and only if, the following determinant is zero:

$$\begin{vmatrix} x_a & x_b & x_c & x_d & x_e \\ y_a & y_b & y_c & y_d & y_e \\ z_a & z_b & z_c & z_d & z_e \\ x_a^2+y_a^2+z_a^2 & x_b^2+y_b^2+z_b^2 & x_c^2+y_c^2+z_c^2 & x_d^2+y_d^2+z_d^2 & x_e^2+y_e^2+z_e^2 \\ 1 & 1 & 1 & 1 & 1 \end{vmatrix} \tag{4}$$

If Determinant (4) is non-zero, then the sign of the determinant and the orientation of the simplex $\Delta(a, b, c, d)$ passing through a, b, c, and d tell us if point e lies in the interior of $\bigcirc(a, b, c, d)$. The orientation of simplex $\Delta(a, b, c, d)$ is again given by a determinant:

$$\begin{vmatrix} x_a & x_b & x_c & x_d \\ y_a & y_b & y_c & y_d \\ z_a & z_b & z_c & z_d \\ 1 & 1 & 1 & 1 \end{vmatrix} \tag{5}$$

If the sign of Determinant (4) equals the sign of Determinant (5), then e lies inside $\bigcirc(a, b, c, d)$.

Plane π passing through a, b, c separates d and e, thus $\Delta(a, b, c, d)$ and $\Delta(a, b, c, e)$ have different orientation. We denote Determinant (4) as $Det(a, b, c, d, e)$. $Det(a, b, c, d, e)$ and $Det(a, b, c, e, d)$ have different sign, because two columns are interchanged. Then, $\bigcirc(a, b, c, d)$ contains e in its interior if, and only if, $\bigcirc(a, b, c, e)$ contains d in its interior. ◀

▶ **Theorem 2.2.** *Let $S = \{a, b, c, d, e\}$ be a set of five points in general and convex position in $\mathbb{R}^3$, such that triangle $\Delta(a, b, c)$ separates d and e, and triangle $\Delta(a, d, e)$ separates b and c. Then, exactly two of the four spheres $\bigcirc(a, b, c, d)$, $\bigcirc(a, b, c, e)$, $\bigcirc(a, d, e, b)$ and $\bigcirc(a, d, e, c)$ contain the remaining point of S in its interior. Furthermore, $\bigcirc(a, b, c, d)$ contains e if, and only if, $\bigcirc(a, b, c, e)$ contains d; and $\bigcirc(a, d, e, b)$ contains c if, and only if, $\bigcirc(a, d, e, c)$ contains b.*

Proof. From Lemma 2.1 we get that $\bigcirc(a, b, c, d)$ contains e if, and only if, $\bigcirc(a, b, c, e)$ contains d; and $\bigcirc(a, d, e, b)$ contains c if, and only if, $\bigcirc(a, d, e, c)$ contains b. The two possible vectors for sphere-point containment for S, since it is in convex position, are $(s_0, s_1) = (3, 2)$ and $(s_0, s_1) = (2, 3)$, corresponding to types B and C. In particular s_0 cannot be $0, 1, 4$ nor 5. Then, exactly two of the four spheres $\bigcirc(a, b, c, d)$, $\bigcirc(a, b, c, e)$, $\bigcirc(a, d, e, b)$ and $\bigcirc(a, d, e, c)$ contain the remaining point of S in its interior. Note that S is of type B or C; this depends on the remaining sphere $\bigcirc(b, c, d, e)$ containing a or not. ◀

▶ Remark. Let $S = \{a, b, c, d, e\}$ be a set of five points in general and convex position in $\mathbb{R}^3$. It follows from Radon's lemma that we can relabel the points from S such that triangle $\Delta(a, b, c)$ separates d and e, and triangle $\Delta(a, d, e)$ separates b and c.

3 A lower bound on the expected number of points in a sphere

▶ **Theorem 3.1.** *For $n \geq 5$,*

$$S_n \geq 2\left\lceil \frac{\binom{n}{5}}{5} \right\rceil + \binom{n}{5} - 2\left\lfloor \frac{n}{25} \right\rfloor,$$

with equality for $5 \leq n \leq 7$. In particular, $S_5 = 1$, $S_6 = 8$, and $S_7 = 29$.

The equality for $n = 5$ is attained for a Type A set of five points. The equalities for $n = 6, 7$ are obtained by the example point sets and by case analysis on the number of points on the convex hull of a generic point set, together with some geometric arguments. For $n > 7$, the inequality follows from Lemmas 3.2, 3.3 and 3.4, and by using induction on n.

▶ **Lemma 3.2.**

$$S_n \geq \left\lceil \frac{n}{n-5} S_{n-1} \right\rceil.$$

▶ **Lemma 3.3.** *For any $n \in \mathbb{N}$, $n \geq 6$, the quotient $\frac{\binom{n-1}{5}}{n-5}$*

(a) is in $\mathbb{N}$, if n is not a multiple of 5,

(b) equals $125\binom{\ell+1}{4} + 25\binom{\ell}{2} + \frac{1}{5}$, if $n = 5\ell$.

▶ **Lemma 3.4.** *The recurrence relation*

$$T(n) = \left\lceil \frac{n}{n-5} T(n-1) \right\rceil \text{ for } n > 7 \text{ and } T(7) = 29,$$

has solution

$$T(n) = 2\left\lceil \frac{\binom{n}{5}}{5} \right\rceil + \binom{n}{5} - 2\left\lfloor \frac{n}{25} \right\rfloor.$$

4 The moment curve and an upper bound

▶ **Lemma 4.1.** *Let S be set of n points on the moment curve $\gamma(t) = (t, t^2, t^3)$, with $t > 0$. Then, all five-tuples of S are of Type B, and $\sum_{k=0}^{n-4} k \cdot s_k = 2\binom{n}{5}$.*

The proof is based on point-in-sphere determinant tests.

▶ **Corollary 4.2.** *Among all sets S of n points in convex and general position in $\mathbb{R}^3$, points on the moment curve minimize $\sum_{k=0}^{n-4} k \cdot s_k$.*

Next, we will use the inversion transformation to construct point sets of arbitrary size in $\mathbb{R}^3$ with all of its five-tuples being Type C. The inversion transformation is determined by two parameters: The center of inversion O and the radius of inversion R. Two points p and p' in $\mathbb{R}^3$ are said to be inverses of each other if:
1. The points p and p' lie in the same half-line with origin in O.
2. The Euclidean distances $|\overline{Op}|$ and $|\overline{Op'}|$ in $\mathbb{R}^3$ satisfy $R^2 = |\overline{Op}||\overline{Op'}|$.

The following is a well-known inversion's property which is key to construct such sets.

▶ **Property 1.** *The inverse of any sphere $\bigcirc$ that does not pass through the center of inversion is a sphere $\bigcirc'$ that also does not pass through the center of inversion. Also, if the center of inversion is in the interior of $\bigcirc$, then the interior of $\bigcirc$ transforms to the exterior of $\bigcirc'$ and the exterior of $\bigcirc$ transforms into the interior of $\bigcirc'$.*

▶ **Theorem 4.3.** *Let S be a set of n points in general position in $\mathbb{R}^3$. Then, $\sum_{k=0}^{n-4} k \cdot s_k \leq 3\binom{n}{5}$ and the bound is tight.*

Proof. The upper bound follows trivially from Equation (3), whose maximum value is reached if all five-tuples are of Type C. Let us then see that the bound is tight.

Let r, s, t, u be four points on the moment curve, such that $0 < r < s < t < u < \frac{1}{10}$. To test if a point $p = (0, y, 0)$ lies inside the sphere $\bigcirc(r, s, t, u)$ passing through r, s, t, u, we examine the sign of the following determinant,

$$\begin{vmatrix} r & s & t & u & 0 \\ r^2 & s^2 & t^2 & u^2 & y \\ r^3 & s^3 & t^3 & u^3 & 0 \\ r^2+r^4+r^6 & s^2+s^4+s^6 & t^2+t^4+t^6 & u^2+u^4+u^6 & y^2 \\ 1 & 1 & 1 & 1 & 1 \end{vmatrix}, \tag{6}$$

that simplifies to $(r-u)(r-t)(r-s)(s-u)(t-u)(s-t) \cdot K$ with

$$\begin{aligned} K = {} & r^3stu + r^2s^2tu + r^2st^2u + r^2stu^2 + rs^3tu + rs^2t^2u + rs^2tu^2 + rst^3u + rst^2u^2 + rstu^3 \\ & + r^3sy + r^3ty + r^3uy + r^2s^2y + 2r^2sty + 2r^2suy + r^2t^2y + 2r^2tuy + r^2u^2y + rs^3y + 2rs^2ty + 2rs^2uy \\ & + 2rst^2y + 3rstuy + 2rsu^2y + rt^3y + 2rt^2uy + 2rtu^2y + ru^3y + s^3ty + s^3uy + s^2t^2y \\ & + 2s^2tuy + s^2u^2y + st^3y + 2st^2uy + 2stu^2y + su^3y + t^3uy + t^2u^2y + tu^3y + rstu + rsy + rty + ruy \\ & + sty + suy + tuy + y^2 - y \end{aligned}$$

being a sum of 50 terms. On one hand, choosing $y = \frac{1}{2}$, we get that each but the last two terms of K are less than $\frac{1}{200}$ and the last two terms are $y^2 - y = -\frac{1}{4}$. Then, $K < 48 \cdot \frac{1}{200} - \frac{1}{4} < 0$. On the other hand, the sign of $(r-u)(r-t)(r-s)(s-u)(t-u)(s-t)$ is positive because of the choice of r, s, t, u. Then Determinant (6) has negative sign. Since the orientation of simplex $\Delta(r, s, t, u)$ is negative, $p = \left(0, \frac{1}{2}, 0\right)$ is contained in the sphere $\bigcirc(r, s, t, u)$. Thus, a set S of arbitrary size can be constructed, using points from the moment curve $\gamma(t) = (t, t^2, t^3)$ with $0 < t < 1/10$, such that every sphere determined by four points in S contains point p.

By Lemma 4.1, all five-tuples of S are of Type B. This implies that for each five-tuple: three of the spheres defined by four of its points do not contain the remaining point, and two of the spheres defined by four of its points contain the remaining point.

Now, by Property 1, since p is contained in the interior of every sphere passing through four points of S, any inversion with center p transforms each Type B five-tuple from S to a Type C five-tuple. Therefore, every five-tuple of the set S' of inverse points from S is of Type C. Thus, from Equation (3), for the set S' we have $\sum_{k=0}^{n-4} k \cdot s_k = 3\binom{n}{5}$. ◀

▶ **Corollary 4.4.** *Among all sets of n points in general position in $\mathbb{R}^3$, $\sum_{k=0}^{n-4} k \cdot s_k$ is maximized for a set S of n points on the curve*

$$\delta(t) = \left(\frac{4t}{4t^6 + 4t^4 + 1}, \frac{4t^6 + 4t^4 + 8t^2 - 3}{8t^6 + 8t^4 + 2}, \frac{4t^3}{4t^6 + 4t^4 + 1} \right)$$

with $0 < t < \frac{1}{10}$. All the five-tuples of S are of Type C.

Note that the curve $\delta(t)$ is the inversion of the moment curve, as described in Theorem 4.3.

5 A universal constant for points in spheres containment

Let S be a set of $n \geq d + 2$ points in general position in $\mathbb{R}^d$. Let $P_d(S)$ be the probability that the sphere passing through $d + 1$ points chosen uniformly at random from S, contains another point chosen uniformly at random from the remaining points of S. We have

$$P_d(S) = \frac{\sum_{k=0}^{n-d-1} k \cdot s_k}{(d+2)\binom{n}{d+2}}. \tag{7}$$

To see this, first observe that there are $\binom{n}{d+2}$ ways to choose $d + 2$ different points from S, and among them, any can be the point to test to be inside or outside the sphere determined by the other $d + 1$ points. On the other hand, for a sphere enclosing k points of S, we count k times a sphere containing another point. Altogether, there are $\sum_{k=0}^{n-d-1} k \cdot s_k$ spheres determined by $d + 1$ points that contain another point from S. All these spheres containing a point are equally likely to be chosen.

We define $P_d(n) = \min P_d(S)$, where the minimum is taken among all sets S of n points in general position in $\mathbb{R}^d$, and $P_d^* = \lim_{n\to\infty} P_d(n)$. We show that this limit exists.

▶ **Lemma 5.1.** *For each dimension $d \geq 2$, there exists a constant $0 \leq P_d^* \leq 1$ such that*

$$P_d^* = \lim_{n\to\infty} P_d(n).$$

From Theorem 3.1 and the proof of Lemma 5.1 we obtain the following corollary:

▶ **Corollary 5.2.**

$$P_3^* \geq \frac{7}{25}.$$

References

1 G. Albers, L. J. Guibas, J. S. B. Mitchell, T. Roos. Voronoi diagrams of moving points. *International Journal of Computational Geometry and Applications* 8(3): 365–379, 1998.

2 J. Akiyama, Y. Ishigami, M. Urabe, J. Urrutia. On circles containing the maximum number of points. *Discrete Mathematics* 151: 15–18, 1996.

3 I. Bárány, H. Schmerl, S. J. Sidney, J. Urrutia. A combinatorial result about points and balls in Euclidean space. *Discrete & Computational Geometry* 4(3): 259–262, 1989.

4 M. de Berg, O. Cheong, M. van Kreveld, M. Overmars. Computational geometry: algorithms and applications. Springer, 2008.

5 M. Claverol, C. Huemer, A. Martínez-Moraian. On circles enclosing many points. *Discrete Mathematics* 344(10): 112541, 2021.

6 J. H. Conway, H. T. Croft, P. Erdős, M. J. T. Guy. On the distribution of values of angles determined by coplanar points. *Journal of the London Mathematical Society.* s2-19(1): 137–143, 1979.

7 B. Delaunay. Sur la sphère vide. *Bulletin de l'Académie des Sciences de l'URSS.* (6):793–800, 1934.

8 R. A. Dwyer. Higher-Dimensional Voronoi Diagrams in Linear Expected Time. *Discrete & Computational Geometry* (6):343–367, 1991.

9 H. Edelsbrunner, A. Garber, M. Saghafian. On spheres with k points inside. arXiv: `https://arxiv.org/abs/2410.21204`, 2024.

10 H. Edelsbrunner, N. Hasan, R. Seidel, X. J. Shen. Circles through two points that always enclose many points. *Geometriae Dedicata* 32: 1–12, 1989.

11 H. Edelsbrunner, F. P. Preparata, D. B. West. Tetrahedrizing point sets in three dimensions. *Journal of Symbolic Computation* 10: 335–347, 1990.

12 R. Fabila-Monroy, C. Huemer, E. Tramuns. The expected number of points in circles. 28th European Workshop on Computational Geometry (EuroCG 2012), 69–72, 2012.

13 J.E. Goodman, R. Pollack. Multidimensional sorting. *SIAM Journal on Computing* 12(3): 484–507, 1983.

14 L. Guibas, J. Stolfi, J. C. Spehner. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics* 4: 74–123, 1985.

15 R. Hayward. A note on the circle containment problem. *Discrete & Computational Geometry* 4: 263–264, 1989.

16 R. Hayward, D. Rappaport, R. Wenger. Some extremal results on circles containing points. *Discrete & Computational Geometry* 4: 253–258, 1989.

17 S. Heubach, N. Y. Li, T. Mansour. Staircase tilings and k-Catalan structures. *Discrete Mathematics* 308(24): 5954–5964, 2008.

18 R. Kahkeshani, A generalization of the Catalan numbers. *Journal of Integer Sequences* 16, Article 13.6.8, 2013.

19 É. Lucas. Sur les congruences des nombres eulériens et des coefficients différentiels des fonctions trigonométriques suivant un module premier. *Bulletin de la Société mathématique de France* 6: 49–54, 1878.

20 R. Meštrović. Lucas' theorem: its generalizations, extensions and applications (1878-2014). `https://arxiv.org/pdf/1409.3820`, arXiv, 2014.

21 V. Neumann-Lara, J. Urrutia. A combinatorial result on points and circles on the plane. *Discrete Mathematics* 69: 173–178, 1988.

22 OEIS Foundation Inc. (2025), The On-Line Encyclopedia of Integer Sequences, Published electronically at `https://oeis.org`.

23 A. Okabe, B. Boots, K. Sugihara, S.N. Chiu. Spatial tessellations: concepts and applications of Voronoi diagrams. Second edition, Wiley, 2000.

24 M. N. Prodromou. A combinatorial property of points and balls, a colored version. *Discrete & Computational Geometry* 38: 641–650, 2007.

25 P. A. Ramos, R. Viaña. Depth of segments and circles through points enclosing many points: a note. *Computational Geometry* 42: 338–341, 2009.

26 T. Roos. Voronoi diagrams over dynamic scenes. *Discrete Applied Mathematics* 43: 243–259, 1993.

27 S. Smorodinsky, M. Sulovský, U. Wagner. On center regions and balls containing many points. COCOON 2008. *Lecture Notes in Computer Science* 5092: 363–373. Springer, 2008.

28 B. L. Rothschild, E. G. Straus. On triangulations of the convex hull of n points. *Combinatorica* 5(2): 167–179, 1985.

29 M. Schaefer. The graph crossing number and its variants: a survey. *The Electronic Journal of Combinatorics*, #DS21, Eighth Edition, 2024.

30 E. R. Scheinerman and H. S. Wilf. The rectilinear crossing number of a complete graph and Sylvester's "Four Point Problem" of geometric probability. *The American Mathematical Monthly* 101(10): 939–943, 1994.

31 J. Urrutia. A containment result on points and circles. Preprint 2004. `https://www.matem.unam.mx/~urrutia/online_papers/PointCirc2.pdf`

Partially Ordered Hamiltonian Paths of Grid Graphs

Jesse Beisegel[1], **Katharina Klost**[2], **Kristin Knorr**[2], **Fabienne Ratajczak**[1], **and Robert Scheffler**[1]

1 **Institut für Mathematik, BTU Cottbus-Senftenberg**
`{jesse.beisegel,fabienne.ratajczak,robert.scheffler}@b-tu.de`

2 **Institut für Informatik, Freie Universität Berlin**
`{katharina.klost,kristin.knorr}@fu-berlin.de`

Abstract

We consider the complexity of finding an ordered Hamiltonian path that obeys a partial order on the vertices in grid graphs and graphs with bounded pathwidth. We show that the problem is NP-complete in $h \times w$ grid graphs for $h \geq 7$, implying NP-completeness for graphs of treewidth and pathwidth at least 7. Contrarily, we give a polynomial-time algorithm for graphs of pathwidth 3.

Related Version `https://arxiv.org/abs/2503.03553`

1 Introduction

For some applications of the well-known Traveling Salesman Problem (TSP) and its path variant it is necessary to add additional precedence constraints to the vertices which ensure that some vertices are visited before others. The cycle variant is known as the Traveling Salesman Problem with Precedence Constraints (TSP-PC) and has been studied, e.g., in [1,6]. The path variant, known as the Sequential Ordering Problem (SOP) or the Minimum Setup Scheduling Problem, has been studied, e.g., in [2,9,11,12]. Of course, all these problems are NP-complete.

These problems are defined over complete graphs with an additional cost function. The unweighted variants Hamiltonian Path and Hamiltonian Cycle with presedence constraints for non-complete graphs have not received the same level of attention for a long time. Results have been only given for the very restricted variants where one or both endpoints of the Hamiltonian path are fixed. For these problems, polynomial-time algorithms have been presented for several graph classes including (proper) interval graphs [3,4,16,17], distance-hereditary graphs [14,18], and rectangular grid graphs [15].

To overcome this lack of research, Beisegel et al. [5] introduced the Partially Ordered Hamiltonian Path Problem (POHPP). The input is a graph $G = (V, E)$ together with a partial order π on V. The question is if there is an ordered Hamiltonian path $(v_1, \dots, v_n)$ in G such that for all $i, j \in \{1, \dots, n\}$ it holds that if $(v_i, v_j) \in \pi$, then $i \leq j$. As POHPP is a generalization of Hamiltonain Path, the problem is also NP-hard. However, Beisegel et al. [5] show that POHPP is already NP-hard for complete bipartite graphs and complete split graphs – graph classes where Hamiltonian Path is trivial.

There are several other classes of graphs, where Hamiltonian Path can be solved in polynomial time but the complexity of POHPP is open. In this work, we consider the POHPP for grid graphs. It is easy to see that every grid graph has a Hamiltonian path, so Hamiltonian Path is trivial for that class. We show, that for grids of heights at least 7 the problem becomes NP-complete, when allowing partial order constraints. We contrast this hardness proof with a polynomial-time algorithm for graphs of pathwidth 3 which directly implies a polynomial-time algorithm for grids of height at most 3.

2 Preliminaries

A $h \times w$ grid graph is a graph with vertex set $\{1, \dots, h\} \times \{1, \dots, w\}$ and edges between each pair of vertices with hamming distance one. We call w the *width* of the graph and h the *height* of the graph. Throughout the paper we assume that the size of the partial order on the vertex set is in $\Omega(w + h)$. Let $\mathcal{P}$ be an ordered Hamiltonian path and π be a partial order. Then $\mathcal{P}$ is a *π-extending Hamiltonian path* if the order of the vertices in $\mathcal{P}$ is a linear extension of π.

Let Φ be a 3-SAT formula with variables $x_1, \dots, x_n$ and clauses $c_1, \dots, c_m$. Denote by $\ell_j^1, \ell_j^2, \ell_j^3$ the literals of clause c_j. We say that ℓ_j^a is a positive literal if it has the form x_i and a negative literal otherwise.

Let G be a graph. Then an ordered set $\mathcal{X} = (X_1, \dots, X_\ell)$ with $X_i \subseteq V$ is a *path decomposition* of G of width w if **1.** $|X_i| \leq w + 1$ for all X_i; **2.** For every edge $e \in E$ there is at least one X_i with $e \subseteq X_i$; **3.** For $i \leq j \leq k$, $X_i \cap X_k \subseteq X_j$. A graph has *pathwidth* at most k if there is a path decomposition of width k for G. It is known that a $h \times w$-grid graph has pathwidth $\min\{w, h\}$ [8].

3 NP-completeness for Grids of Height at Least 7

▶ **Theorem 3.1.** *POHPP is* NP*-complete on grid graphs of height* 7*.*

Proof. We present a reduction from 3-SAT to POHPP. Let Φ be a 3-SAT formula with n variables and m clauses as defined above. We construct an instance (G, π) for POHPP where G is a $7 \times (5n + 4m + 6)$ grid graph. G is conceptually subdivided into different gadgets, described below. We also name some special vertices within these gadgets. See Figure 1 for an illustration. For a gadget Z, we use $Z[a, b]$ to denote the vertex in the a-th row and b-th column of the gadget.

Start gadget S (rose): A 7×3 grid graph. We denote the vertex $S[3, 3]$ by s.
Variable switch gadgets Y_i (blue): A 7×1 grid graph for $i = 1, \dots n$. Y_i is assigned to variable x_i.
Variable gadget X_i (yellow): A 7×4 grid graph for $i = 1, \dots, n$. X_i is assigned to variable x_i. We call the vertices $X_i[3, 2]$ and $X_i[3, 3]$ the *negative variable vertices* of X_i. The vertices $X_i[5, 2]$ and $X_i[5, 3]$ are the *positive variable vertices.*
Middle gadget M (red): A 7×2 grid graph.
Clause switch gadget D_j (light blue): A 7×2 grid graph D_j assigned to c_j for $j = 1, \dots, m$.
Clause gadget C_j (green): A 7×2 grid graph assigned to c_j for $j = 1, \dots m$. The vertices $C_j[a + 2, 1]$ and $C_j[a + 2, 2]$ are assigned to the literal ℓ_j^a.
End gadget T (rose): A 7×1 grid graph. We denote the vertex $T[5, 1]$ by t.

G is made up of the gadgets in the following order: $S, Y_1, X_1, \dots, Y_n, X_n, M, D_1, C_1, \dots, D_m, C_m, T$. The partial order π gives the following constraints:

1. $s \prec v$ for all $v \in G$.
2. $t \prec v$ for all $v \in S \setminus \{s\}$.
3. $t \prec v$ for all vertices v in Rows $1, 2, 6$ and 7.
4. $t \prec v$ if v is in Row 4 of some X_i
5. $u \prec v$ if u is a negative variable vertex in X_i and v in C_j is assigned to a literal $\overline{x}_i$
6. $u \prec v$ if u is a positive variable vertex in X_i and v in C_j is assigned to a literal x_i.

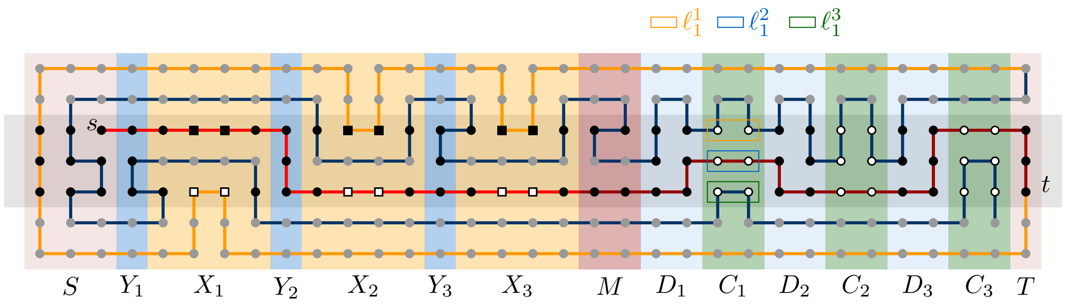


Figure 1 The gadgets for Theorem 3.1 combined to represent a formula. The gray vertices come after t in the partial order π. A black square is a negative variable vertex, a white square is a positive variable vertex. The white dot vertices are the clause vertices. These are assigned to the literals als indicated.

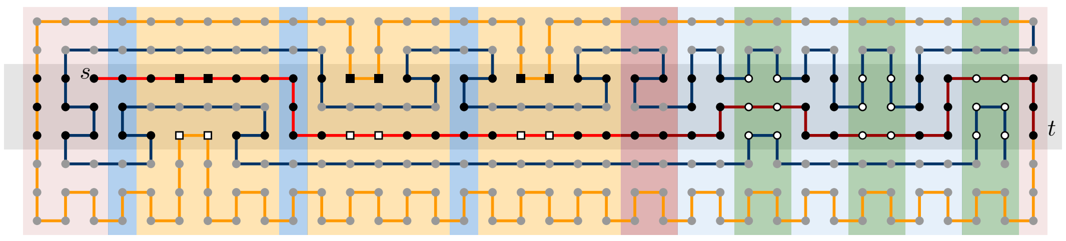


Figure 2 Example for the construction with height 8.

We call the last two constraints *literal constraints*. Now we show that the instance described above has a π-extending Hamiltonian path if and only if the given formula Φ is satisfiable.

Let $\mathcal{P}$ be a π-extending Hamiltonian path of G. By the first constraint, $\mathcal{P}$ starts in s. Let $\mathcal{P}_{s,t}$ be the prefix of $\mathcal{P}$ ending in t. Observe that by the definition of π, all vertices that are in Rows $1, 2, 6$ or 7 or in Row 4 of any variable gadget cannot lie on $\mathcal{P}_{s,t}$. This implies that for any X_i, the path $\mathcal{P}_{s,t}$ either contains only the positive or only the negative variable vertices. We define an assignment of the variables of Φ as follows. If $\mathcal{P}_{s,t}$ traverses X_i through the negative variable vertices, we set $x_i = 0$, otherwise we set $x_i = 1$. The literal constraints ensure that this assignment is satisfying as a clause gadget can only be crossed in $\mathcal{P}_{s,t}$ if at least one literal of the associated clause is satisfied.

For the other direction, assume that there is a satisfying assignment of the variables. The path starts in s and visits the negative variable vertices for all variables assigned 0 and the positive variable vertices for the remaining variables, using the variable switch gadgets to reach the appropriate rows. Then each clause gadget is crossed in the vertices that are assigned to one true literal in the clause, using the clause switch gadgets to get to the appropriate row, going up in the second and down in the first column of the gadget. Then the remaining vertices are visited as indicated in Figure 1 and Figure 3. ◀

▶ **Corollary 3.2.** *POHPP is* NP*-complete on grid graphs of height h for $h \geq 7$.*

Proof. The construction in Theorem 3.1 can be extended as indicated in Figure 2. The added vertices in the lower rows take over the constraints of the vertices in row 7. ◀

▶ **Corollary 3.3.** *POHPP is* NP*-hard for graphs of pathwidth and treewidth ≥ 7.*

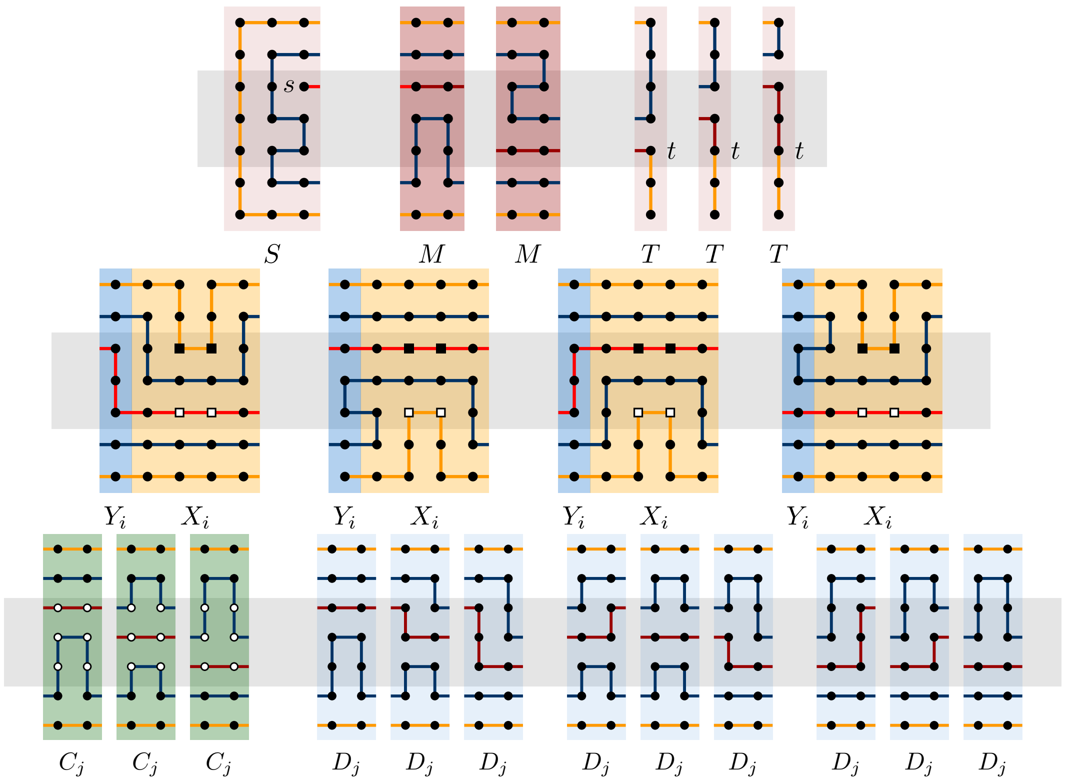


Figure 3 The path $\mathcal{P}$ for the different kinds of gadgets.

4 Polynomial-Time Algorithm for Pathwidth 3

In this section, we contrast Theorem 3.1 with a polynomial-time algorithm for graphs of pathwidth at most 3. Let $\mathcal{X} = (X_1, \ldots, X_k)$ be a path decomposition of width 3 of a graph G. For our algorithm, we assume that all bags have size exactly four and that X_i and X_{i+1} differ in exactly one vertex. We call the unique vertex $u \in X_i \setminus X_{i+1}$ the vertex that is *forgotten* in X_{i+1} and the unique vertex $w \in X_{i+1} \setminus X_i$ the vertex that is *introduced* in X_{i+1}. Furthermore, let $V_i = \bigcup_{j=1}^{i} X_j$. We can compute a general path-decomposition of width 3 in linear time [7, 13]. It is easy to see that such a path decomposition can be modified in linear time to fulfill our conditions above.

▶ **Observation 4.1.** *The vertex u forgotten in X_{i+1} has no edge to a vertex in $V \setminus V_i$. The vertex w introduced in X_{i+1} has no edge to a vertex in $V_i \setminus X_i$.*

Our algorithm is based on the folklore dynamic programming algorithm that solves Hamiltonian Path for graphs of bounded pathwidth which we will not describe here due to space constraints.[1] Let $\mathcal{P}$ be an ordered Hamiltonian path and let $\mathcal{P}_i = V_i \cap \mathcal{P}$ be the set of subpaths of $\mathcal{P}$ that is induced by V_i. Each $P = (v_1, \ldots, v_\ell) \in \mathcal{P}_i$ is either a suffix of $\mathcal{P}$, a prefix of $\mathcal{P}$, a midpart of $\mathcal{P}$ or, in the case $k = i$, the complete path $\mathcal{P}$. We call v_1 and v_ℓ the start or end vertex of P, respectively. Vertices that are not the start or end vertices are called

[1] There seems to be no description of this algorithm available; it is asked for in Exercise 7.19 of the textbook of Cygan et al. [10]. A short description of the algorithm for Hamiltonian Cycle is given by Ziobro and Pilipczuk [19].

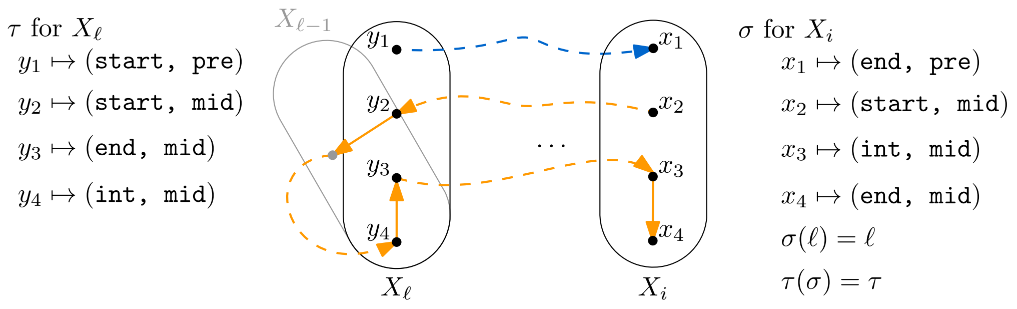


Figure 4 Illustration of the definition of a signature for a fixed directed Hamiltonian path $\mathcal{P}$. We have $\mathcal{P}(\sigma) = \{$`pre`, `mid`$\}$

interior vertices. In the dynamic program for the HAMILTONIAN PATH problem without order restrictions, it suffices to store the "interface" between X_{i-1} and X_i. As a solution for POHPP has to consider π, we also need information about the partition of the vertices in X_i to the paths. In general, there are exponentially many of these partitions, making this approach infeasible. For pathwidth 3, however, we can show that this information can be maintained and computed efficiently. The following lemma is the basis for our algorithm.

▶ **Lemma 4.2.** *For $1 \leq i < k$, $\mathcal{P}_i$ has one of the following forms:* ***1.*** *$\mathcal{P}_i$ contains exactly one non-trivial path;* ***2.*** *$\mathcal{P}_i$ contains a non-trivial prefix and a suffix;* ***3.*** *$\mathcal{P}_i$ contains a non-trivial midpart and a non-trivial prefix or a suffix.*

The following lemma helps us to reduce the number of possible path partitions by fixing the bag where a second path appears.

▶ **Lemma 4.3.** *For each path $\mathcal{P}$ such that $\mathcal{P}_i$ contains a midpart and a prefix (or suffix), there is an index $\ell \leq i$ such that all $\mathcal{P}_j$ for $j = \ell, \ldots, i$ contain a midpart and a prefix (or suffix) and either $\ell = 1$ or $\mathcal{P}_{\ell-1}$ contains only one non-trivial path.*

We define the *signature* σ of a bag X_i as a mapping of the vertices in X_i to the possible types of vertices and paths. If σ induces a midpart and another path, then σ also contains a value $\ell(\sigma)$ for $1 \leq \ell(\sigma) \leq i$. If $\ell(\sigma) \neq i$, it also contains a signature $\tau(\sigma)$ that induces a midpart and another parth for $X_{\ell(\sigma)}$ with $\ell(\tau) = \ell(\sigma)$. Let $\mathcal{P}(\sigma)$ be the set types of paths induced by σ. See Figure 1 for an illustration of this concept. We call a signature *valid* if $\mathcal{P}(\sigma)$ follows the form of Lemma 4.2 and the mapping of the vertex types is consistent with the paths. Intuitively, a signature σ encodes how a path that is a candidate for a π-extending Hamiltonian path interacts with X_i and in the case of a midpart and another path, how the second path appeared.

We say that signatures γ, σ for X_{i-1} and X_i are *compatible* if they are both valid and there is a way to extend the paths induced by γ with the vertex w introduced in X_i to get the signature σ. If σ induces a midpart and another path and $\ell(\sigma) < i$, then σ and γ are only compatible if $\ell(\sigma) = \ell(\gamma)$ and $\tau(\sigma) = \tau(\gamma)$ for $\ell(\sigma) \leq i - 2$ and $\tau(\sigma) = \gamma = \tau(\gamma)$ if $\ell(\sigma) = i - 1$. Furthermore, if $\ell(\sigma) = i$, then a signature γ is only compatible to σ if γ induces only one non-trivial path.

Given a signature σ for X_i, a *path mapping* f_σ assigns each vertex $v \in V_i$ to one of the paths induced by σ. If σ induces a midpart and another path, then f_σ partitions the vertices in the way described by Lemma 4.3. This path mapping is uniquely determined for $i = 1$ and if $\mathcal{P}(\sigma)$ contains only one non-trivial path, or $\ell(\sigma) = i$. If $\mathcal{P}(\sigma)$ contains a midpart and another path, we additionally require that f_σ is consistent with the unique path mapping

$f_{\tau(\sigma)}$. A path mapping *contradicts* π if there is no possible order in which the vertices in $V \setminus V_i$ can be appended and prepended to the paths without violating the constraints given by π, e.g., if a vertex in the prefix has some predecessor in π that is contained in $V \setminus V_i$.

▶ **Lemma 4.4.** *Let σ be a signature for X_i such that $\mathcal{P}(\sigma)$ contains a non-trivial midpart and another non-trivial path. Then there is at most one signature γ for X_{i+1} that is compatible to σ such that $\mathcal{P}(\gamma)$ contains a non-trivial midpart and another non-trivial path.*

Proof. This statement follows from Lemma 4.3 and the structure of $\mathcal{X}$. ◀

Now we can define the dynamic program that will solve POHPP: For each bag X_i and each valid signature σ for X_i, define the subproblem $D[i, \sigma]$. If there is an an exact path mapping f_σ for X_i that does not contradict π then $D[i, \sigma] = f_\sigma$, otherwise $D[i, \sigma] = \perp$. The algorithm considers increasing values of i. In the base case $D[1, \sigma]$ there is only one possible path partition f_σ. If it does not contradict π, then set $D[1, \sigma] = f_\sigma$ and otherwise set $D[1, \sigma] = \perp$.

For an entry $D[i, \sigma]$ with $i \geq 2$, iterate over all valid signatures γ for X_{i-1} that are compatible to σ and where $D[i-1, \gamma] \neq \perp$. Let w be the vertex introduced in X_i and f_σ^γ the path partition that extends $D[i-1, \gamma]$ by assigning w to the path defined by σ. As σ and γ are consistent, the position of w in the paths is uniquely determined. Iterate over the vertices $V \setminus \{w\}$ to explicitly check if adding w in this unique position contradicts π.

If there is a signature γ such that w can be added to the path, then set $D[i, \sigma] = f_\sigma^\gamma$. In the other case set $D[i, \sigma] = \perp$. If there is a valid signature σ for X_k such that $D[k, \sigma] \neq \perp$, then the algorithm returns *yes*, in the other case, it returns *no*.

▶ **Theorem 4.5.** *POHPP in graphs of pathwidth at most 3 can be solved in $O(n^3)$ time.*

Proof. Lemma 4.4 directly implies that if $\mathcal{P}(\sigma)$ contains a midpart and another path, there is only one compatible signature γ with $D[i-1, \gamma] \neq \perp$. The running time follows directly. For the correctness, one has to argue that storing one path partition is always enough. If σ induces only one non-trivial path or a midpart and another path, by Lemmas 4.3 and 4.4 there is only one possible path partition to consider. If it induces a prefix and a suffix, then the vertices in $V \setminus V_i$ are added between the end vertex of the prefix and the start vertex of the suffix. Thus, if there are multiple path partitions for σ that do not contradict π they only differ in vertices that are incomparable to vertices in $V \setminus V_i$ and it does not matter which one is stored. ◀

▶ **Corollary 4.6.** *POHPP in grids of height at most 3 can be solved in $O(n^3)$ time.*

5 Conclusion

We showed that POHPP is NP-complete for grids of height at least 7. On the other hand there is a polynomial-time algorithm for grids of height at most 3. It would be interesting to close the gap in future work.

References

1 Zakir Hussain Ahmed and S. N. Narahari Pandit. The travelling salesman problem with precedence constraints. *Opsearch*, 38(3):299–318, 2001. doi:10.1007/BF03398638.

2 Norbert Ascheuer, Laureano F. Escudero, Martin Grötschel, and Mechthild Stoer. A cutting plane approach to the sequential ordering problem (with applications to job scheduling in manufacturing). *SIAM Journal on Optimization*, 3(1):25–42, 1993. doi:10.1137/0803002.

3 Katerina Asdre and Stavros D. Nikolopoulos. The 1-fixed-endpoint path cover problem is polynomial on interval graphs. *Algorithmica*, 58(3):679–710, 2010. doi:10.1007/s00453-009-9292-5.

4 Katerina Asdre and Stavros D. Nikolopoulos. A polynomial solution to the k-fixed-endpoint path cover problem on proper interval graphs. *Theoretical Computer Science*, 411(7):967–975, 2010. doi:10.1016/j.tcs.2009.11.003.

5 Jesse Beisegel, Fabienne Ratajczak, and Robert Scheffler. Computing hamiltonian paths with partial order restrictions. *ACM Transactions on Computation Theory*, 17(1), 2025. doi:10.1145/3711844.

6 Lucio Bianco, Aristide Mingozzi, Salvatore Ricciardelli, and Massimo Spadoni. Exact and heuristic procedures for the traveling salesman problem with precedence constraints, based on dynamic programming. *INFOR: Information Systems and Operational Research*, 32(1):19–32, 1994. doi:10.1080/03155986.1994.11732235.

7 Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996. doi:10.1137/S0097539793251219.

8 Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.

9 Charles J. Colbourn and William R. Pulleyblank. Minimizing setups in ordered sets of fixed width. *Order*, 1:225–229, 1985. doi:10.1007/BF00383598.

10 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, Cham, 2015. doi:10.1007/978-3-319-21275-3.

11 Laureano F. Escudero. An inexact algorithm for the sequential ordering problem. *European Journal of Operational Research*, 37(2):236–249, 1988. doi:10.1016/0377-2217(88)90333-5.

12 Laureano F. Escudero. On the implementation of an algorithm for improving a solution to the sequential ordering problem. *Trabajos de Investigacion-Operativa*, 3:117–140, 1988.

13 Martin Fürer. Faster computation of path-width. In Veli Mäkinen, Simon J. Puglisi, and Leena Salmela, editors, *Combinatorial Algorithms - 27th International Workshop, IWOCA 2016*, volume 9843 of *LNCS*, pages 385–396. Springer, 2016. doi:10.1007/978-3-319-44543-4_30.

14 Sun-Yuan Hsieh. An efficient parallel strategy for the two-fixed-endpoint hamiltonian path problem on distance-hereditary graphs. *Journal of Parallel and Distributed Computing*, 64(5):662–685, 2004. doi:10.1016/j.jpdc.2004.03.014.

15 Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982. doi:10.1137/0211056.

16 Peng Li and Yaokun Wu. A linear time algorithm for the 1-fixed-endpoint path cover problem on interval graphs. *SIAM Journal on Discrete Mathematics*, 31(1):210–239, 2017. doi:10.1137/140981265.

17 George B. Mertzios and Walter Unger. An optimal algorithm for the k-fixed-endpoint path cover on proper interval graphs. *Mathematics in Computer Science.*, 3(1):85–96, 2010. doi:10.1007/s11786-009-0004-y.

18 Hong-Gwa Yeh and Gerard J. Chang. The path-partition problem in bipartite distance-hereditary graphs. *Taiwanese Journal of Mathematics*, 2(3):353–360, 1998. doi:10.11650/twjm/1500406975.

19 Michal Ziobro and Marcin Pilipczuk. Finding hamiltonian cycle in graphs of bounded treewidth: Experimental evaluation. *ACM J. Exp. Algorithmics*, 24(1):2.7:1–2.7:18, 2019. doi:10.1145/3368631.

Segment Watchman Routes*

Anna Brötzner[1], Omrit Filtser[2], Bengt J. Nilsson[1], Christian Rieck[3], and Christiane Schmidt[4]

1 **Department of Computer Science and Media Technology, Malmö University, Sweden**
`{anna.brotzner, bengt.nilsson.TS}@mau.se`

2 **Department of Mathematics and Computer Science, The Open University of Israel, Isreal**
`omrit.filtser@gmail.com`

3 **Department of Discrete Mathematics, University of Kassel, Germany**
`christian.rieck@mathematik.uni-kassel.de`

4 **Department of Science and Technology, Linköping University, Sweden**
`christiane.schmidt@liu.se`

Abstract

We consider a variant of the 2-watchmen problem that ensures that every point in a polygon $\mathbf{P}$ is seen from more than one direction: we search for routes W_1, W_2, such that for each $p \in \mathbf{P}$ there exist $w_1 \in W_1, w_2 \in W_2$ that see p and such that $p \in \overline{w_1 w_2} \subset \mathbf{P}$. We show that finding the two routes that are optimal with respect to the min-max criterion is NP-hard in simple polygons and present a 2-approximation algorithm for this case; moreover, we provide a polynomial-time algorithm for computing the two optimal routes with respect to the min-sum criterion in convex polygons. Finally, we discuss a generalized version of the problem with more than two watchmen.

1 Introduction

In the classical Watchman Route Problem, introduced by Chin and Ntafos [3, 4], we ask for the shortest route inside a given simple polygon $\mathbf{P}$, such that all points of $\mathbf{P}$ are visible from at least one point on the route (this can be solved in polynomial time [14, 15]). In this context, a point $p \in \mathbf{P}$ *sees* another point $q \in \mathbf{P}$ if the line segment $\overline{pq}$ is fully contained in $\mathbf{P}$.

Carlsson et al. [2] raised the m-watchmen problem as a natural generalization: we are given m watchmen (with or without given starting points) for which we aim to find routes, such that each point in $\mathbf{P}$ is visible from at least one of the m routes. Two common objectives for this problem are to minimize the total length of all m watchman routes (called *min-sum*) and to minimize the length of the longest route assigned to any watchman (called *min-max*).

When considering m watchmen, we only require each point to be seen at least once, without any guarantees on any kind of robustness. However, in practice, we may aim to make our routes robust against potential issues. For example, one or more watchmen may fail, especially in remote regions. Additionally, observing a point from multiple angles can improve observation quality. This is crucial to make the theoretically intriguing routes applicable for real-world scenarios. In this paper, we aim to enhance the coverage quality by guaranteeing a point to be seen from multiple directions.

* A. B., B. J. N., and C. S. are supported by grant 2021-03810 (Illuminate: provably good algorithms for guarding problems) from the Swedish Research Council (Vetenskapsrådet). O. F. is supported by the Israel Science Foundation (grant no. 2135/24). C. R. is partially supported by the German Research Foundation (DFG) grant no. 522790373.

Problem Definition. Let $\mathbf{P}$ be a *simple polygonal domain.* A route W in $\mathbf{P}$ is called a *watchman route* if every point in $\mathbf{P}$ is visible from some point on W, and we denote its length by $|W|$. A point $p \in \mathbf{P}$ is *segment-guarded* by two points $w_1, w_2 \in \mathbf{P}$ if p lies on the line segment $\overline{w_1 w_2}$, and p is visible to both w_1 and w_2, i.e., $\overline{w_1 w_2}$ is fully contained in $\mathbf{P}$ (while the watchmen do not need to be at w_1 and w_2 at the same time).

Two routes W_1, W_2 in $\mathbf{P}$ are *segment watchman routes* for $\mathbf{P}$ if for every point $p \in \mathbf{P}$ there exist two points $w_1 \in W_1$ and $w_2 \in W_2$ such that p is segment-guarded by w_1 and w_2. We consider the following two problems:

▷ Problem 1 (Min-Max Segment Watchmen). Given a polygonal domain $\mathbf{P}$, find segment watchman routes W_1, W_2 such that $\max_i |W_i|$ is minimized.

▷ Problem 2 (Min-Sum Segment Watchmen). Given a polygonal domain $\mathbf{P}$, find segment watchman routes W_1, W_2 such that $\sum_i |W_i|$ is minimized.

In the same manner, we define a point $p \in \mathbf{P}$ to be *triangle-guarded* (or *k-gon-guarded*) if there exist points w_i on routes W_i, $i = 1, 2, 3$ (or $i = 1, \ldots, k$), such that the segments $\overline{w_i p}, \forall i$, are fully contained in $\mathbf{P}$ and do not share a point other than p. With this, we define the related min-max and min-sum optimization problems analogously to Problems 1 and 2.

Note that, due to limited space, we omit the proofs of statements marked by $(\star)$.

Related Work. Carlsson et al. [2] showed that the m-watchmen problem is NP-hard in simple polygons and provided a polynomial time algorithm for histograms. Polynomial time algorithms for different polygon classes, using either the min-sum or the min-max objective, have also been presented in [1, 9, 11, 12]. Recently, Nilsson and Packer [10] proposed a 5.969-approximation algorithm to compute min-max 2-watchman routes in simple polygons.

The robustness requirement we employ for watchman routes in this paper is closely related to the problems of two-sensor visibility and triangle guarding for stationary guards introduced by Efrat et al. [6] and Smith and Evans [13], respectively. Both considered two polygons $\mathbf{Q}, \mathbf{P}$ with $\mathbf{Q} \subseteq \mathbf{P}$, where the subpolygon $\mathbf{Q}$ should be guarded by guards placed in $\mathbf{P}$ (assuming that $\mathbf{Q}$'s boundary is transparent). For Efrat et al. a point $p \in \mathbf{Q}$ is *2-guarded at angle α* by two guards g_1, g_2 if $\angle g_1 p g_2 \in [\alpha, \pi - \alpha]$ and both guards see p. Smith and Evans defined a point $p \in \mathbf{Q}$ to be *triangle-guarded* by g_1, g_2, g_3 if p is seen by each of the three guards and is contained in the triangle spanned by them. Another variant of robust guarding has recently been established by Das et al. [5]; and a variant of robustness for a single watchman by Langetepe et al. [8].

2 Preliminaries and Key Lemma

Let $\mathbf{P}$ be a simple polygon with n vertices. We assume that $\mathbf{P}$ does not contain vertices with an internal angle of exactly 180°, i.e., no three consecutive vertices are on the same line. If $\mathbf{P}$ does contain such a vertex, we can simply remove it.

Let W_1, W_2 be segment watchman routes for $\mathbf{P}$. From the definition, we obtain:

▶ **Observation 2.1.** *Each of W_1 and W_2 is a watchman route for $\mathbf{P}$.*

▷ Claim 2.2. Every convex vertex of $\mathbf{P}$ is visited by one of W_1 or W_2.

Proof. Let v be a convex vertex of $\mathbf{P}$. Then v lies on a line segment $\overline{w_1 w_2}$ with $w_1 \in W_1$ and $w_2 \in W_2$, and the segments $\overline{v w_1}, \overline{v w_2}$ are contained in $\mathbf{P}$. As the interior angle at v is strictly smaller than 180°, any line segment in $\mathbf{P}$ that contains v has v as one of its endpoints. ◁

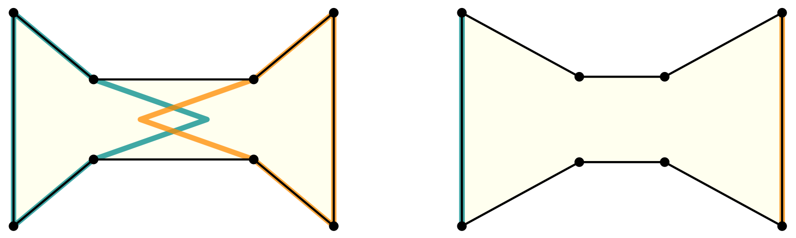


Figure 1 Min-max segment watchman routes may or may not need to overlap.

We now establish sufficient conditions for two routes to be segment watchman routes; an example is illustrated in Figure 1.

▶ **Lemma 2.3** (The Conditions Lemma). *Two routes W_1 and W_2 are segment watchman routes for $\mathbf{P}$ if the following conditions hold:*

1. *Every convex vertex is visited by one of W_1 or W_2.*
2. *Both W_1 and W_2 visit the visibility polygon of each convex vertex.*
3. *Both W_1 and W_2 are simple and relatively convex (i.e., a route does not cross itself, and for any two points inside the region enclosed by the route, their shortest path is also contained within the enclosed region).*

Proof. First, we show that Condition 2 implies that W_1 and W_2 are watchman routes. Assume that there is a point $p \in \mathbf{P}$ that is not seen by W_i, i.e., no point of W_i lies in p's visibility polygon. Hence, W_i is fully contained in one of the *pockets* $\mathbf{P}'$ of p's visibility polygon (a subpolygon of $\mathbf{P}$ in which no point is visible from p). Extend the pocket's *window* w (the line segment that separates $\mathbf{P}'$ and $\mathbf{P} \setminus \mathbf{P}'$) into a maximal line segment ℓ contained in $\mathbf{P}$. Without loss of generality, let ℓ be a vertical line segment with $\mathbf{P}'$ to its right. As $p \in \ell$, $\ell \setminus w$ is either a polygonal edge with a convex endpoint not seen by W_i, or it splits $\mathbf{P}$ into at least two subpolygons; see Figure 2. At least one of the subpolygons, say $\mathbf{P}''$, also lies to the right of ℓ. W_i cannot see any convex vertex in $\mathbf{P}''$, yielding a contradiction.

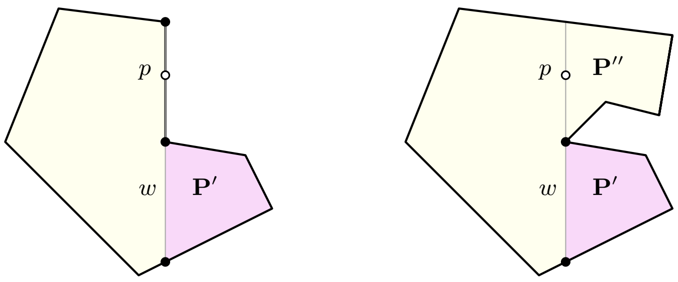


Figure 2 $\ell \setminus w$ is either an edge of $\mathbf{P}$ with a convex endpoint (left), or it splits $\mathbf{P}$ into at least two subpolygons, one of which also lies to the right of ℓ (right).

We now show that Conditions 1–3 imply that W_1 and W_2 are segment watchman routes. Consider a point $p \in \mathbf{P}$. Since both W_1 and W_2 are watchman routes, there exists at least one point on W_1 and at least one point on W_2 that p sees. Consider the two wedges defined by the angles from which p is viewing W_1 and W_2, as visualized in Figure 3: let F_1 be the maximal wedge bounded by two rays starting at p, such that for every ray ρ in F_1 there is a point $w \in W_1$ in this direction that p sees. Note that because $\mathbf{P}$ is simple, F_1 is a single wedge. The wedge F_2 is defined analogously for W_2.

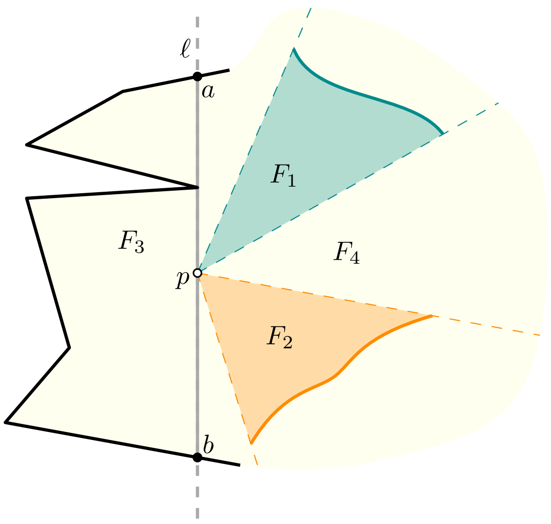


Figure 3 The wedges F_1 and F_2 define the angles from which p is viewing W_1 and W_2, respectively. If F_3 or F_4 is larger than $180°$, then there is a convex vertex on the left side of ℓ which is not visited.

Each of the two wedges F_1, F_2 covers either $360°$ (if p lies on or within the relatively convex route) or less than $180°$ (because both routes are relatively convex). If at least one of F_1, F_2 covers $360°$ around p, then p is segment-guarded: assume that F_2 covers $360°$ around p, and let w_1 be a point on W_1 that sees p. Then the ray from w_1 in the direction of p intersects W_2 at point w_2 that sees p, and thus p is segment-guarded by $\overline{w_1w_2}$.

Hence, assume that neither F_1 nor F_2 covers $360°$ around p. Let F_3 (and possibly F_4) be the maximal wedge(s) bounded by two rays starting at p, such that for every ray ρ in F_3 (and F_4) there is no point $w \in W_1$ or $w \in W_2$ in this direction that p sees. Then the plane around p can be split into up to four wedges, depending on whether F_1 and F_2 intersect: F_1, F_2, F_3 and F_4; or F_1, F_2, F_3, and one wedge with the overlap of F_1 and F_2.

We argue that neither F_3 nor F_4 can cover more than $180°$. Without loss of generality, assume that F_3 covers more than $180°$. Consider a line ℓ through p in F_3 that does not contain an edge of the boundary of $\mathbf{P}$, and assume that ℓ is a vertical line and that both F_1 and F_2 are on the right side of ℓ. Let $\overline{ab}$ be the maximal line segment on ℓ that is contained in $\mathbf{P}$. Then $\overline{ab}$ splits $\mathbf{P}$ into at least two subpolygons, and at least one of them, $\mathbf{P}'$, is on the left side of $\overline{ab}$. Because $\mathbf{P}$ is simple and both W_1 and W_2 do not cross $\overline{ab}$, there are no points of W_1 and W_2 in $\mathbf{P}'$. However, $\mathbf{P}'$ must contain a convex vertex v. This yields a contradiction, as by Condition 1, v needs to be visited by at least one of the watchman routes. ◀

We define the *relative convex hull* of a route in a simple polygon $\mathbf{P}$ as the simple polygon $\mathbf{Q}$ such that, for any two points inside the region enclosed by the route, the geodesic connecting them is also contained within $\mathbf{Q}$. Specifically, we refer to the boundary of $\mathbf{Q}$ as the relative convex hull. Hence, if a route is relatively convex, it coincides with its relative convex hull.

In Lemma 2.3, the conditions imply that W_1 and W_2 are segment watchman routes. However, there exist segment watchman routes that do not fulfill these conditions, see Figure 4.

On the other hand, we obtain an if-and-only-if statement for optimal watchman routes:

▶ **Observation 2.4.** *Let $\mathbf{P}$ be a simple polygon. Two routes W_1 and W_2 are optimal segment watchman routes for $\mathbf{P}$, if and only if the conditions from Lemma 2.3 hold.*

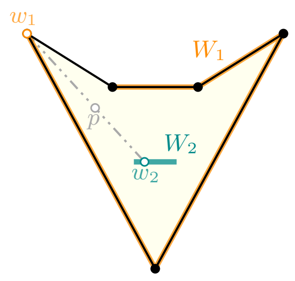


Figure 4 W_1 and W_2 are segment watchman routes (e.g., p lies on $\overline{w_1 w_2}$), but do not fulfill the conditions of Lemma 2.3. They are not optimal, e.g., W_1's relative convex hull (in this case the boundary of the polygon) is shorter than W_1, and this relative convex hull together with W_2 are segment watchman routes.

3 Min-Max Segment Watchman Routes in Simple Polygons

We sketch a reduction showing that the problem is NP-hard even in simple polygons. Complementarily, we provide a polynomial-time 2-approximation.

3.1 Computational Complexity

We reduce from MULTIWAY NUMBER PARTITIONING [7]. In particular, for our purposes, we ask to partition a set of numbers into two sets of equal sum; also referred to as PARTITION, which is known to be weakly NP-hard.

▶ **Theorem 3.1** ($\star$). *Problem 1 is* NP*-hard even in simple polygons.*

Proof sketch. Construct a star-shaped polygon as in Figure 5. The length of a spike's boundary (i.e., the path $v_{2i-1}, v_{2i}, v_{2i+1}$) represents the value α_i from the PARTITION instance φ, and let T denote the sum of all values. Both watchmen start in the bottommost convex vertex v_0, and thus need to return to it. It is easy to see that a min-max segment watchman route of length $T/2 + \varepsilon$ exists iff there exist a partition of φ into two sets of equal sum. ◀

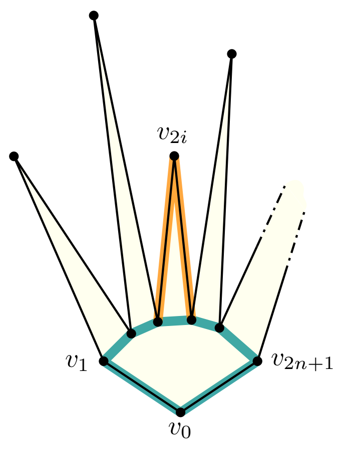


Figure 5 High-level idea of the type of polygon utilized in the NP-hardness reduction.

3.2 Approximation Algorithm

Let k be the number of convex vertices of a given polygon $\mathbf{P}$. We enumerate the convex vertices in counterclockwise order $v_0, \ldots, v_{k-1}$, with v_0 chosen arbitrarily. In the following, we assume, without loss of generality, that indices are counted modulo k.

Let v_i and v_j be two different convex vertices and let C_{ij} be the shortest route that visits the convex vertices $v_i, \ldots, v_{j-1}$. C_{ji} is then the shortest route that visits the convex vertices $v_j, \ldots, v_{i-1}$. Clearly, C_{ij} and C_{ji} can be computed in linear time. Let $C_{\mathbf{P}}$ be the shortest route that visits all the convex vertices of $\mathbf{P}$. $C_{\mathbf{P}}$ can also be computed in linear time.

Let D_{ij} be the shortest route that starts and ends at v_i, and that sees all the convex vertices $v_j, \ldots, v_{i-1}$. The route D_{ji} is then the shortest route that starts and ends at v_j, and that sees all the convex vertices $v_i, \ldots, v_{j-1}$. Each of D_{ij} and D_{ji} can be computed in $\mathcal{O}(n^3)$ time by modifying the algorithm of Jiang and Tan [15]. Let $D_{\mathbf{P}}$ be the shortest floating watchman route in $\mathbf{P}$ (that is, the shortest watchman route without a given starting point). We can compute $D_{\mathbf{P}}$ in $\mathcal{O}(n^4)$ time [14, 15].

Let $\mathrm{RCH}(T)$ denote the relative convex hull of a route T in $\mathbf{P}$. We define $W_{ij} \overset{\mathrm{def}}{=} \mathrm{RCH}(C_{ij} \cup D_{ij})$, connecting the two routes at v_i and taking the relative convex hull of them.

We construct our approximate solution by choosing the pair

$$(W_1, W_2) = \operatorname*{arg\,min}_{i \neq j} \big\{ \max\{|W_{ij}|, |W_{ji}|\}, \max\{|C_{\mathbf{P}}|, |D_{\mathbf{P}}|\} \big\}.$$

By Lemma 2.3, (W_1, W_2) is a feasible solution for the segment watchman routes problem. Denote by $\mathrm{OPT}(\mathbf{P})$ the size of an optimal solution for $\mathbf{P}$. We claim the following result.

▶ **Theorem 3.2.** $\max\big\{|W_1|, |W_2|\big\} \leq 2 \cdot \mathrm{OPT}(\mathbf{P})$.

Proof. Let W_1^* and W_2^* be two segment watchman routes with $\max\big\{|W_1^*|, |W_2^*|\big\} = \mathrm{OPT}(\mathbf{P})$. Without loss of generality, we may assume that W_1^* and W_2^* are as short as possible.

If W_1^* or W_2^* visits all convex vertices of $\mathbf{P}$, then $(C_{\mathbf{P}}, D_{\mathbf{P}})$ is an optimal solution to the problem and the theorem therefore holds. Hence, for the remainder of this proof, we assume that W_1^* visits some fixed convex vertex v_i and W_2^* visits a different fixed convex vertex v_j.

Since W_1^* visits v_i and it either sees or visits the convex vertices $v_j, \ldots, v_{i-1}$ by construction, we have that $|D_{ij}| \leq |W_1^*|$. Similarly, W_2^* visits v_j and it either sees or visits the convex vertices $v_i, \ldots, v_{j-1}$, yielding $|D_{ji}| \leq |W_2^*|$. We distinguish the following cases.

W_1^* and W_2^* do not intersect. Because W_1^* and W_2^* do not intersect, the two convex vertices v_i and v_j can be chosen so that W_1^* visits $v_i, \ldots, v_{j-1}$ by increasing index (modulo k) and sees the remaining ones, whereas W_2^* visits $v_j, \ldots, v_{i-1}$ and sees the remaining ones. From this, it follows that $|C_{ij}| \leq |W_1^*|$ and $|C_{ji}| \leq |W_2^*|$. We obtain that

$$\begin{aligned} \max\big\{|W_1|, |W_2|\big\} &\leq \max\big\{|\mathrm{RCH}(C_{ij} \cup D_{ij})|, |\mathrm{RCH}(C_{ji} \cup D_{ji})|\big\} \\ &\leq \max\big\{2|W_1^*|, 2|W_2^*|\big\} = 2 \cdot \max\big\{|W_1^*|, |W_2^*|\big\} = 2 \cdot \mathrm{OPT}(\mathbf{P}). \end{aligned}$$

W_1^* and W_2^* intersect. Because W_1^* and W_2^* intersect and together visit all the vertices, we have $|C_{\mathbf{P}}| \leq |W_1 \cup W_2| = |W_1^*| + |W_2^*|$ and $|D_{\mathbf{P}}| \leq \min\{|W_1^*|, |W_2^*|\}$, as both W_1^* and W_2^* are watchman routes. We obtain that

$$\begin{aligned} \max\big\{|W_1|, |W_2|\big\} \leq \max\big\{|C_{\mathbf{P}}|, |D_{\mathbf{P}}|\big\} &\leq \max\big\{|W_1^*| + |W_2^*|, \min\{|W_1^*|, |W_2^*|\}\big\} \\ &\leq 2 \cdot \max\big\{|W_1^*|, |W_2^*|\big\} = 2 \cdot \mathrm{OPT}(\mathbf{P}). \end{aligned}$$ ◀

In fact, we may also let $W_2 = C_{\mathbf{P}}$ to avoid computing a floating shortest watchman route. The proof also gives a 2-approximation if we use the min-sum measure for the two routes.

4 Min-Sum Segment Watchman Routes in Convex Polygons

We examine the min-sum variant of the segment watchman routes problem in convex polygons.

▶ **Lemma 4.1** (⋆). *For convex polygons, each of the two optimal min-sum segment watchman routes visits a consecutive set of convex vertices.*

▶ **Corollary 4.2.** *Problem 2 can be solved in polynomial time in convex polygons.*

5 Conclusion and Future Work

In this abstract, we investigated segment watchman routes in simple polygons. We identified sufficient conditions for two watchman routes to be segment watchman routes, and developed a 2-approximation algorithm for the min-max and the min-sum measure. Furthermore, we argued that the problem of computing min-max segment watchman routes for simple polygons is NP-hard, and concluded that computing min-sum segment watchman routes for convex polygons is possible in polynomial time. We plan to extend the study of Problem 2 to general simple polygons.

The NP-hardness of Problem 1 for three and k watchmen follows easily from an adaption of the proof of Theorem 3.1. We aim to investigate these two problems for $k > 2$ in the future.

References

1 Alireza Bagheri, Anna Brötzner, Faezeh Farivar, Rahmat Ghasemi, Fatemeh Keshavarz-Kohjerdi, Erik Krohn, Bengt J. Nilsson, and Christiane Schmidt. Minsum m watchmen's routes in Stiegl polygons. In *XX Spanish Meeting on Computational Geometry*, 2023.

2 Svante Carlsson, Bengt J. Nilsson, and Simeon C. Ntafos. Optimum guard covers and m-watchmen routes for restricted polygons. *International Journal of Computational Geometry and Applications*, 3(1):85–105, 1993. doi:10.1007/BFb0028276.

3 Wei-Pang Chin and Simeon C. Ntafos. Optimum watchman routes. In *Symposium on Computational Geometry (SoCG)*, pages 24–33, 1986. doi:10.1145/10515.10518.

4 Wei-Pang Chin and Simeon C. Ntafos. Optimum watchman routes. *Information Processing Letters*, 28(1):39–44, 1988. doi:10.1016/0020-0190(88)90141-X.

5 Rathish Das, Omrit Filtser, Matthew J. Katz, and Joseph S. B. Mitchell. Robustly Guarding Polygons. In *Symposium on Computational Geometry (SoCG)*, pages 47:1–47:17, 2024. doi:10.4230/LIPIcs.SoCG.2024.47.

6 Alon Efrat, Sariel Har-Peled, and Joseph S. B. Mitchell. Approximation algorithms for two optimal location problems in sensor networks. In *International Conference on Broadband Networks (ICBN)*, pages 714–723, 2005. doi:10.1109/ICBN.2005.1589677.

7 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

8 Elmar Langetepe, Bengt J. Nilsson, and Eli Packer. Discrete surveillance tours in polygonal domains. In *Canadian Conference on Computational Geometry (CCCG)*, 2017. URL: https://www.cccg.ca/proceedings/2017/CCCG2017.pdf.

9 Joseph S. B. Mitchell and Erik L. Wynters. Watchman routes for multiple guards. In *Canadian Conference on Computational Geometry (CCCG)*, pages 293–327, 1991. URL: http://cccg.ca/proceedings/1991/paper31.pdf.

10 Bengt J. Nilsson and Eli Packer. Approximation algorithms for the two-watchman route in a simple polygon. *Algorithmica*, 86(9):2845–2884, 2024. doi:10.1007/s00453-024-01245-0.

11 Bengt J. Nilsson and Sven Schuierer. Shortest m-watchmen routes for histograms: The minmax case. In *International Conference on Computing and Information (ICCI)*, pages 30–33, 1992. `doi:10.1109/ICCI.1992.227712`.

12 Bengt J. Nilsson and Derick Wood. Optimum watchmen routes in spiral polygons: Extended Abstract. In *Canadian Conference on Computational Geometry (CCCG)*, pages 269–272, 1990. URL: `https://cccg.ca/proceedings/1990/Paper58.pdf`.

13 Jocelyn Smith and William S. Evans. Triangle guarding. In *Canadian Conference on Computational Geometry (CCCG)*, pages 76–80, 2003. URL: `https://www.cccg.ca/proceedings/2003/46.pdf`.

14 Xuehou Tan. Fast computation of shortest watchman routes in simple polygons. *Information Processing Letters*, 77(1):27–33, 2001. `doi:10.1016/S0020-0190(00)00146-0`.

15 Xuehou Tan and Bo Jiang. Efficient algorithms for touring a sequence of convex polygons and related problems. In *Theory and Applications of Models of Computation (TAMC)*, pages 614–627, 2017. `doi:10.1007/978-3-319-55911-7_44`.

Two Watchmen's Routes in Staircase Polygons*

Anna Brötzner[1], Bengt J. Nilsson[1], and Christiane Schmidt[2]

1 **Department of Computer Science and Media Technology, Malmö University, Sweden**
`{anna.brotzner,bengt.nilsson.TS}@mau.se`

2 **Department of Science and Technology, Linköping University, Sweden**
`christiane.schmidt@liu.se`

Abstract

We consider the watchman route problem for multiple watchmen in staircase polygons, which are rectilinear x- and y-monotone polygons. For two watchmen, we propose an optimal algorithm that takes quadratic time, improving on the cubic time of the trivial solution. For $m \geq 3$ watchmen, we explain where our approach fails.

1 Introduction

The watchman route problem asks for a shortest route inside a polygon, such that every point in the polygon is visible to some point on the route. It was first introduced by Chin and Ntafos [2], who showed that the problem is NP-hard for polygons with holes, but may be solved efficiently for simple polygons. Given a starting point, an optimal route can be computed in $O(n^3)$ time [8], and finding a solution without a fixed starting point takes a linear factor longer [7].

The Watchman Route Problem has also been considered for multiple watchmen (a problem introduced by Carlsson, Nilsson, and Ntafos [1]). For histograms, efficient algorithms have been proposed for minimizing the total route length (min-sum) [1] and the length of the longest route (min-max) [6]. Here, we are interested in only two watchmen. For this problem, Mitchell and Wynters [4] proved NP-hardness for the min-max objective in simple polygons. Recently, Nilsson and Packer presented a polynomial-time 5.969-approximation algorithm for the same objective in simple polygons [5].

In this paper, we consider a quite restricted class of polygons, staircase polygons, that for two watchmen allows us to assign the responsibility for guarding any edge solely to one of the two watchmen (and seeing all of a polygon's boundary is for two watchmen sufficient to see the polygon). Additionally, we show that the two routes can be separated by a diagonal between two reflex vertices. This enables a polynomial-time algorithm to compute the optimal two watchman routes (for both the min-max and the min-sum objective). Despite staircase polygons being so restricted, some of the observations we make do not hold for three or more watchmen. This indicates a discrepancy in the computational complexity between the watchman route problem for one or two watchmen and for multiple watchmen.

2 Notation and Preliminaries

A polygon is called *rectilinear* (or *orthogonal*) if all its edges are parallel to the x- or the y-axis of a given coordinate system, and *x-monotone* (*y-monotone*) if every line that is

* A. B., B. J. N. and C. S. are supported by grant 2021-03810 (Illuminate: provably good algorithms for guarding problems) from the Swedish Research Council (Vetenskapsrådet).

orthogonal to the x-axis (y-axis) intersects the polygon in exactly one connected interval. A *staircase polygon* is a rectilinear polygon that is both x- and y-monotone. We call the polygonal chains of boundary edges that lie above and below the interior the *ceiling* and the *floor* of the polygon, respectively. We consider the watchman route problem for multiple watchmen in staircase polygons.

▶ Multiple Watchman Route Problem (m-WRP). Given a polygon P, and a number of watchmen m, find a shortest set of m routes, with respect to the min-sum or min-max criterion, such that every point in P is seen from at least one of the routes.

We denote the length of a route w by $\|w\|$, and refer to a solution of the m-WRP as a set of m watchman routes in P. For simplicity, we will refer to a watchman routes also as a watchman. In the following, we consider the m-WRP for the min-sum and the min-max criterion. Any statement on optimal watchman routes holds for either objective, unless stated otherwise.

Let P be a staircase polygon that is not 2-guardable with point guards (as then none of the watchmen would need to walk). As P is x- and y-monotone, we make the following observation:

▶ **Observation 2.1.** *A watchman w with leftmost x-coordinate $x_{\min}$ and rightmost x-coordinate $x_{\max}$ sees at least all points $p \in P$ with $x(p) \in [x_{\min}, x_{\max}]$.*

The analogous statement holds for the bottommost y-coordinate $y_{\min}$ and the topmost y-coordinate $y_{\max}$ of watchman w. Watchman w thus sees the contiguous part of the ceiling between $y_{\min}$ and $x_{\max}$, and the contiguous part of the floor between $x_{\min}$ and $y_{\max}$.

We denote the extensions of edges that are incident to reflex vertices as *cuts*, and identify so-called *essential cuts*. For a single watchman, a simple polygon is seen if all its essential cuts are visited. Clearly, visiting all essential cuts is a necessary condition for a set of watchman routes. A staircase polygon has at most four essential cuts (see Figure 1(a)): the leftmost vertical extension of the floor v_{left}, the lowest horizontal extension of the ceiling h_{bot}, the rightmost vertical extension of the ceiling v_{right}, and the topmost horizontal extension of the floor h_{top}. By "visiting" such an extension, we mean that a watchman route has a point to the left of v_{left}, below h_{bot}, to the right of v_{right}, or above h_{top}. Note that not necessarily all of these four extensions are essential cuts. For the sake of simplicity, we will nevertheless refer to them as such.

For one watchman, an optimal solution is given by the shortest route that visits all four essential cuts. An example is shown in Figure 1(a). By the following theorem proven by Chin and Ntafos [2], such a solution may be computed in linear time.

▶ **Theorem 2.2.** *(Theorem 2 [2], Chin & Ntafos) A shortest watchman route in simple rectilinear polygons can be found in linear time.*

For multiple watchman routes, the watchmen share the responsibility of seeing P. Thus, we aim to find a "good" distribution of responsibilities among the watchmen. For two watchmen, we prove that the polygon may be split into two subpolygons such that an optimal solution to the 2-WRP corresponds to an optimal solution to the WRP in each subpolygon.

3 Computing an Optimal Solution for Two Watchmen

In this section, we investigate the 2-WRP. Let us first state some properties of two optimal watchman routes in staircase polygons. Due to limited space, we omit the proof of Lemma 3.1.

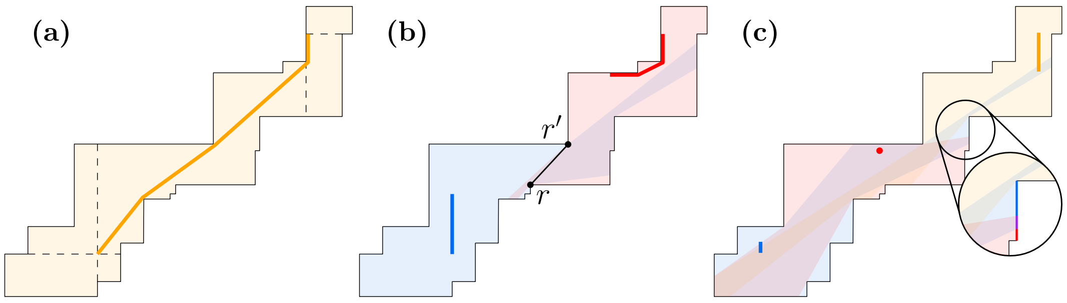


Figure 1 Optimal solutions for (a) one watchman, (b) two watchmen, (c) three watchmen.

▶ **Lemma 3.1.** *Let (w_1^*, w_2^*) be an optimal solution to the 2-WRP in a staircase polygon P. Then, the following properties hold:*

1. *w_1^* and w_2^* do not have any common x- and y-coordinate.*
2. *w_1^* visits the essential cuts h_{bot}, v_{left}, and w_2^* visits the essential cuts h_{top}, v_{right}.*
3. *There exists a pair of reflex vertices (r, r') with r on the floor and r' on the ceiling, such that $\overline{rr'}$ separates w_1^* and w_2^*, see Figure 1(b).*

In the following, we always assume that an optimal solution (w_1^*, w_2^*) obeys Properties 1–3 of Lemma 3.1. In particular, w_1^* lies below and to the left of w_2^*.

▶ **Lemma 3.2.** *In an optimal solution to the 2-WRP in a staircase polygon, for every polygon edge there exists a watchman that sees the edge completely.*

Proof. Let (w_1^*, w_2^*) be an optimal solution, and consider w_1^*. As soon as it crosses the extension of a horizontal floor edge e, it sees e completely since nothing blocks the visibility between w_1^* and e along e's extension. Similarly, w_1^* sees a vertical edge on the ceiling completely as soon as it crosses the edge's extension. Before crossing the extension, w_1^* does not see the respective edge at all. Hence, for any horizontal floor edge (vertical ceiling edge) e, if w_1^* sees any point on e, then it sees all points of e. Similarly, for any horizontal ceiling edge (vertical floor edge) e, if w_2^* sees any point on e, then it sees all points of e. Assume w.l.o.g. that there is a horizontal floor edge e such that no point on e is seen by w_1^*. Then, w_2^* sees e completely as otherwise there are points on e that are not seen by any of w_1^* and w_2^*. ◀

With this, we may split two optimal watchman routes in a particular way.

▶ **Lemma 3.3.** *Let (w_1^*, w_2^*) be an optimal solution in a staircase polygon P. There exists a unique diagonal between a vertex on the floor and a vertex on the ceiling that cuts P into two subpolygons P_1 and P_2 such that w_1^* sees P_1, and w_2^* sees P_2.*

Proof. By Lemma 3.2, every edge is completely seen by a watchman. For a chain of consecutive edges on the floor or ceiling, there cannot be an alteration in the responsibility of the watchmen: Let e_i, e_{i+1}, e_{i+2} be three consecutive edges (on the floor or ceiling). If one watchman sees e_i, e_{i+2} completely, then it also sees e_{i+1}. Hence, there exist vertices on the floor and the ceiling such that w_1^* sees all edges that lie below and to the left of them completely, and w_2^* sees all edges that lie above and to the right of them completely. We call such vertices *breaking points* and show that there exist two breaking points, one on the floor and one on the ceiling, that see each other—these define the unique diagonal. Assume that this is not the case. Let b_f be the lowest-leftmost breaking point on the floor, and b_c be the upper-rightmost breaking point on the ceiling. W.l.o.g., assume that all breaking points on the floor lie to the upper-right of the breaking points on the ceiling (in particular, b_f lies to the upper-right of b_c).

Since b_f and b_c do not see each other, there exist some edges incident to a reflex vertex r that block the visibility. Assume that these edges lie on the ceiling. Then, the horizontal edge incident to r lies above b_c and below b_f, and is seen by w_2^* (by definition of b_c). Hence, w_2^* sees the vertical floor edge v that is hit by the horizontal extension through r (as described in the proof of Lemma 3.2), and thereby also sees the convex vertex on the lower end of v, contradicting the choice of b_f (being the lowest-leftmost breaking point on the floor). ◀

We present an algorithm that finds an optimal split, and thus computes an optimal solution for two watchmen in $O(n^2)$ time. Observe that Lemma 3.2 only holds for two watchmen. For three or more watchmen, some edges may only be seen partially by each watchman in an optimal solution. An example is shown in Figure 1(c), see the magnified part. The blue watchman is in charge of monitoring a part of a vertical floor edge above the red watchman's visibility region. The yellow watchman does not see this edge at all and would have to walk very far to reach the vertical extension of this edge. Therefore, an optimal solution for $m \geq 3$ watchmen may induce a split of the polygon's floor and ceiling into more than m parts each, such that every part is seen by a single watchman. This means that a watchman may be "in charge of" more than one contiguous part of the boundary on the floor and ceiling, respectively.

3.1 A Quadratic-Time Algorithm for Two Watchmen

To compute an optimal solution, because of Lemma 3.3, we consider all diagonals between vertices on the floor and on the ceiling. Any such diagonal splits P into two subpolygons. For each subpolygon, we compute an optimal watchman route using a modified version of the linear-time algorithm proposed by Chin and Ntafos [2], and then combine the two routes to a solution for the 2-WRP in P.

As there are at most quadratically many diagonals to consider, this procedure trivially yields a cubic-time algorithm. However, maintaining a similar structure of the subpolygons by dealing with the diagonals in a certain order allows us to compute many of the watchman routes in amortized constant time.

To this end, we iterate over the vertices on the floor. For each floor vertex p_f, we compute all its diagonals to points on the ceiling, in clockwise order around p_f. If p_f is a convex vertex, then all diagonals have a negative slope. If p_f is a reflex vertex, some diagonals have positive slope. However, we do not need to consider all diagonals with positive slope, but only those two that are followed or preceded by a positive-slope diagonal in the clockwise order. We call those, and the diagonals with negative slopes, *candidate diagonals*; see Figure 3. Every candidate diagonal splits P into two subpolygons, P_1 below and P_2 above the diagonal.

▶ **Lemma 3.4.** *Any diagonal that is not a candidate diagonal induces a solution that is at least as long as the solution induced by some candidate diagonal.*

Proof. First, note that a diagonal of positive slope is spanned between two reflex vertices. Consider w.l.o.g. a non-candidate diagonal $\overline{p_f p_c}$, as seen in Figure 2. Then there is a convex vertex p_c' above p_c that does not yield a diagonal of p_f because $y(p_c) < y(p_f)$. The subpolygon P_2 above $\overline{p_f p_c}$ has the horizontal line through p_c' as an essential cut. Hence, the watchman route in P_2 has points below this cut. There exists a subpolygon induced by a candidate diagonal (incident to p_c and with the other endpoint p_f' below p_f) that also has the horizontal line through p_c' as an essential cut. For this cut, the watchman route in the subpolygon

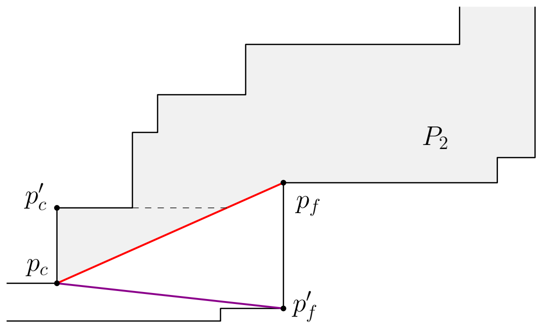


Figure 2 A diagonal with positive slope (red) that is not a candidate: An optimal watchman route in the subpolygon P_2 (marked in gray) needs to visit the same essential cut (dashed line) as an optimal watchman route in the subpolygon induced by $\overline{p_c p'_f}$ (purple).

above the diagonal $\overline{p_c p'_f}$ remains the same, and the watchman route in the subpolygon below is not longer than the one induced by $\overline{p_f p_c}$. ◀

Now we compute a solution for each candidate diagonal in the following manner:

- Step 1: Consider a diagonal with negative slope. Cutting along this diagonal creates only convex vertices in each subpolygon, hence all four essential cuts per subpolygon are rectilinear. The watchman routes touch these extensions, but do not cross them [2]. We compute the optimal solutions for the subpolygons induced by the first diagonal in clockwise order in linear time by Theorem 2.2. In addition, we compute two shortest-path-tree data structures [3]. One is rooted at the first reflex vertex on the floor and stores the shortest paths to all other floor vertices, the other one is rooted at the first reflex vertex on the ceiling and stores the shortest paths to all other ceiling vertices.
  Then, for each diagonal in order, we update the solution in the following way. Moving from one diagonal to the next (i.e., moving from one vertex on the ceiling to the next) alters either the essential cut $v_{\text{right}}(P_1)$ of P_1, or the essential cut $h_{\text{bot}}(P_2)$ of P_2. During this movement, any reflex vertex on the ceiling that was an anchor point can only be released once per vertex p_f, and they are released from right to left. Similarly, any reflex vertex on the floor can be added as an anchor point only once per vertex p_f, and they get added from left to right. Hence, the number of updates per vertex p_f is at most linear. When updating the route w_1 in P_1, we move from one vertical extension $v_{\text{right}}(P_1)$ to the next one $v'_{\text{right}}(P_1)$. We use the shortest-path-tree of the floor to check whether vertices on the floor get added to, and the shortest-path-tree of the ceiling to check whether vertices on the ceiling get released from the route. This can be done in amortized constant time [3].
- Step 2: If p_f is a reflex vertex, we need to consider also the two candidate diagonals with positive slope. Here, the subpolygons' essential cuts differ from those of a staircase polygon: There is exactly one non-rectilinear essential cut, namely the extension of the diagonal. We may nevertheless compute an optimal solution, using the algorithm by Chin and Ntafos [2]. This algorithm defines a set of essential cuts, along which the polygon is reflected. Computing the shortest path from one of these essential cuts to its copy yields the shortest watchman route in the original polygon. Since there are at most five essential cuts, we can try all combinations of subsegments of these essential cuts and apply the Chin-and-Ntafos reduction which takes linear time in each of these constant number of cases.

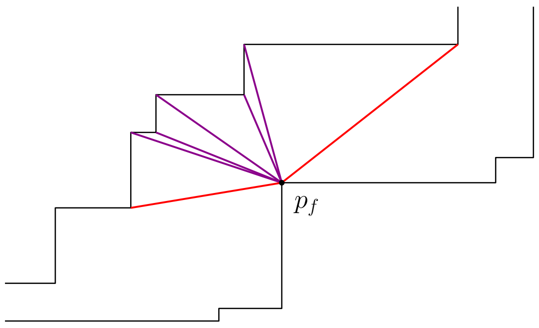


Figure 3 The candidate diagonals of a reflex vertex p_f: there are two candidate diagonals with positive slope (red), and several candidate diagonals with negative slope (purple).

Thus, the computations for each vertex p_f take amortized linear time. As we do this for every vertex on the floor, there are linearly many vertices to consider. With this, we get an optimal solution to the 2-watchman route problem in staircase polygons.

▶ **Theorem 3.5.** *An optimal solution to the* 2*-WRP in staircase polygons can be computed in* $O(n^2)$ *time.*

References

1 Svante Carlsson, Bengt J. Nilsson, and Simeon C. Ntafos. Optimum guard covers and m-watchmen routes for restricted polygons. *Int. J. Comput. Geom. Appl.*, 3(1):85–105, 1993. doi:10.1142/S0218195993000063.

2 Wei-Pang Chin and Simeon C. Ntafos. Optimum watchman routes. *Inf. Process. Lett.*, 28(1):39–44, 1988. doi:10.1016/0020-0190(88)90141-X.

3 D. Harel and R.E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13(2):338–355, 1984.

4 Joseph S. B. Mitchell and Erik L. Wynters. Watchman routes for multiple guards. In *Proc. 3rd CCCG*, volume 9, pages 293–327, 1991. URL: http://cccg.ca/proceedings/1991/paper31.pdf.

5 Bengt J. Nilsson and Eli Packer. Approximation algorithms for the two-watchman route in a simple polygon. *Algorithmica*, 86(9):2845–2884, Sep 2024. doi:10.1007/s00453-024-01245-0.

6 Bengt J. Nilsson and Sven Schuierer. Shortest m-watchmen routes for histograms: the minmax case. In *Proceedings ICCI '92: Fourth International Conference on Computing and Information*, pages 30–33, 1992. doi:10.1109/ICCI.1992.227712.

7 Xuehou Tan. Fast computation of shortest watchman routes in simple polygons. *Information Processing Letters*, 77(1):27–33, 2001. doi:10.1016/S0020-0190(00)00146-0.

8 Xuehou Tan and Bo Jiang. Efficient algorithms for touring a sequence of convex polygons and related problems. In T. V. Gopal, Gerhard Jäger, and Silvia Steila, editors, *Theory and Applications of Models of Computation - 14th Annual Conference, 2017, Bern, Switzerland, April 20-22, 2017, Proceedings*, volume 10185 of *LNCS*, pages 614–627, 2017. doi:10.1007/978-3-319-55911-7_44.

Assembly Order Planning for Modular Structures by Autonomous Multi-Robot Systems

Kenneth Cheung[1], Irina Kostitsyna[2], and Tom Peters[3]

1 Coded Structures Lab, NASA Ames Research Center, USA
2 KBR, Inc. Coded Structures Lab, NASA Ames Research Center, USA
3 TU Eindhoven, The Netherlands t.peters1@tue.nl

Abstract

Coordinated multi-agent robotic construction provides a means to build infrastructure in extreme environments. Planning methods are important to understanding and achieving the scope of such applications. Here, we focus on the NASA Automated Reconfigurable Mission Adaptive Digital Assembly Systems (ARMADAS) model, which includes passive lightweight structural modules and small robots that traverse the structure. We present an algorithm for calculating a build plan for robots under the constraints of this type of system. We evaluate the quality of this plan experimentally. Many of the techniques we use can be applied to any robotic assembly system whose robots perform locomotion over the structure that they are building.

Related Version A video explaining the algorithm is available at `https://www.youtube.com/watch?v=-rzTm3RqIF0`.

1 Introduction

Hybrid programmable matter describes systems for autonomous construction that employ multiple robotic agents and passive construction materials. In these systems, a primary structure is composed of matter that does not perform movement on its own, but must be acted upon by distinct robotic agents. Robots in these systems are smaller than the structures they can build, and the construction matter can have features that allow structure parts to fixture to each other and to help them align during placement [1]. A natural robot type for walking over fixtured systems are inchworm type robots [2, 3]. These robots allow for a wide range of operations using only low numbers of degrees of freedom.

Across robot types, it has been investigated what structures can be built and in what order. Reinforcement learning has been considered [4]. Mixed integer linear programming was used to generate an optimal build plan [5]. This was later sped up by first deconstructing the target structure into independent substructures that can be planned individually and separate from each other [6], and subsequently improved to use a hierarchical subdivision of the structure [7]. It is also of interest to determine what conditions are required for a structure to be buildable. Using a model where cubic modules slide around and are not allowed to disconnect the structure during construction, it has been shown that all structures lacking certain forbidden patterns can be built [8].

Here, we address the Automated Reconfigurable Mission Adaptive Digital Assembly Systems (ARMADAS) infrastructure. The building blocks are cuboctahedron voxels that connect to each other into a cubic grid. The system uses two types of robots [12]. The first type is called MMIC-I [10]: it crawls through the structure of voxels and bolts them together. The second is called SOLL-E [9]: an inchworm type robot with two feet and a so-called backpack with which it can carry voxels. It can move over a construction of voxels by performing inchworm type movements, see Figure 1.

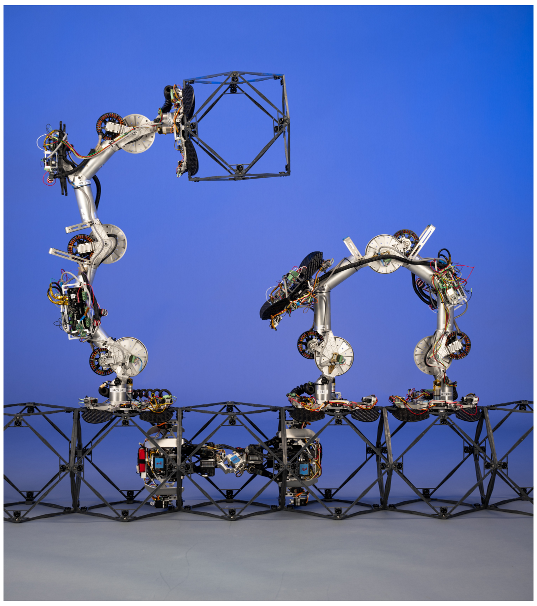


Figure 1 A SOLL-E [9] picking up a voxel from the backpack of another SOLL-E. A MMIC-I [10] is present inside the voxels [11].

These robots can work together to create a structure C out of the voxels. We assume that there is a prebuilt plane *base* of voxels that the robots can utilize to move. The initial voxels of C can be bolted to this base. Furthermore, we assume that there exists a *depot*: a single spot on which SOLL-E's get a voxel loaded on their backpack.

Prior descriptions of ARMADAS system implementations demonstrated teams of three robots. A *cargo* SOLL-E, a *crane* SOLL-E, and a *bolter* MMIC-I. The crane and the bolter robot go to where a new voxel needs to be attached, while the cargo robot goes to the depot first to pick up a voxel. The cargo robot then goes to the construction site, where the crane picks up the voxel from the backpack and holds it in the correct place. Then, the MMIC-I bolts it in place.

To enable precise alignment, voxels have mechanical alignment features, see Figure 2. These bumps and corresponding dents allow for more precision and more stability in the final structure. However, they protrude slightly outside the bounding box of a single voxel, and hence new voxels need to move away before being able to slide into place. This, along with normal deformities under the weight of the structure as well as the robots, might make it impossible for a voxel to get into the correct position. In particular, it is impossible for a voxel to pass through a unit-wide gap between two other voxels. We call this the *no-tight-building* constraint.

This constraint imposes a direct ordering on all connected voxels in a row as soon as one of them has been placed. This significantly limits the possible options to build different parts of the structure at the same time if the algorithm starts with a suboptimal voxel.

2 Build order

Every voxel that is being placed can subsequently be walked on by the robots. However, it also limits their movement and might make the path to some areas longer, or even block off certain areas entirely. Therefore, the order in which voxels are placed can heavily impact

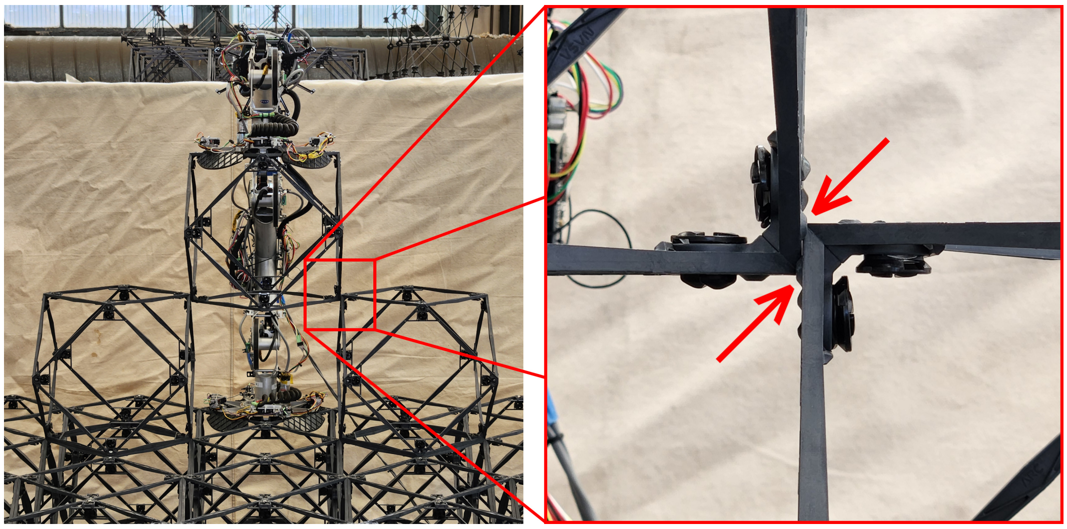


Figure 2 [8] Photograph of armadas robot attempting to place a voxel between two other voxels and failing due to collisions of mechanical alignment features (red arrows).

the total construction time, or even make it impossible to finish the construction. Moreover, the no-tight-building constraint imposes a complete order on all connected voxels in a row or column. Lastly, when robots operate close to each other, they can block each other. Hence, if two teams of robots need to place voxels right next to each other, it might be slower than letting a single robot place both voxels. To minimize the total assembly time, we want to increase parallelism by letting robots operate on independent parts of the same structure.

Algorithm. We assume that the configuration C to be built is connected. We create a directed acyclic dependency graph G_C of configuration C that adheres to the no-tight-building constraint. To construct G_C, we first create a *slice graph* of C [13] as follows. For each z-coordinate, we split the voxels into the connected components per layer. For each of these connected components, we create a single node v representing the voxels in that component. With slight abuse of notation, we say a voxel c is in v ($c \in v$) if c is contained in the connected component represented by v. We insert a directed edge (u, v) for two nodes u and v, if and only if there exist two voxels $c_1 \in u$ and $c_2 \in v$ such that c_2 is directly above c_1, see Figure 3. We use this slice graph as a base for G_C.

Let U be the set of vertices not reachable from a root containing voxels on the ground. We take the vertex $v \in U$ with highest z-coordinate such that there exists a reachable vertex u for which v is a predecessor. We make u a predecessor of v instead, and hence remove v from U. We repeat this until U is empty. The resulting graph is G_C, see Figure 3.

▶ **Lemma 2.1.** *If the assembly order within each node of G_C adheres to the no-tight-building constraint, then the order imposed by G_C adheres to the same constraint.*

Lastly we simplify G_C by contracting edges. We repeatedly take an edge (u, v) such that v is the only successor of u, and u is the only predecessor of v, and *contract* it; we replace both u and v with a new node v' representing the voxels of both u and v. In essence, these two nodes represent a "pillar" in the configuration without any branching, see Figure 4.

The resulting dependency graph G_C ensures that robots can work on nodes that are mutually independent. To order the voxels inside a node v, we run a breadth-first search

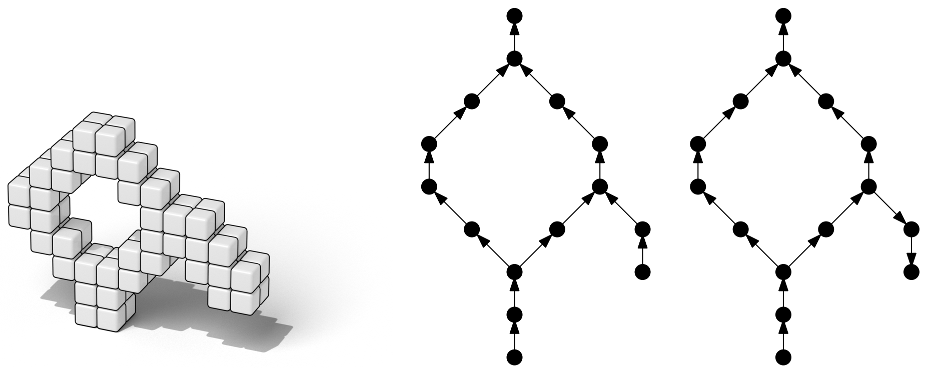


Figure 3 A configuration C, its corresponding slice graph, and its corresponding slice graph with all nodes reachable from a ground root by having some dependencies reversed.

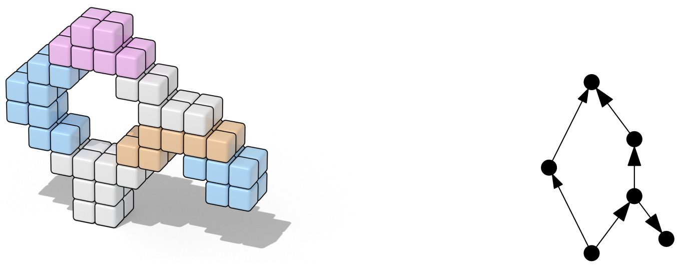


Figure 4 The compressed dependency graph G_C of the configuration shown in Figure 3, and that same configuration with each of the vertices of the nodes of G_C highlighted in a different color.

on all voxels in v, starting from any voxel in v that neighbors a voxel in a predecessor of v. We call the resulting graph G_v. To fix the potential violations of the no-tight-building constraint within G_v, we repeatedly pick a voxel x that violates the constraint and flip it's edges such that it has no more incoming edges from a voxel with x-, y-, or z-coordinate higher than it. If multiple voxels x exist that violate the constraint, we pick x to be the voxel with minimum z-, then y-, then x-coordinate. If the flipping of these edges makes another voxel violate the no-tight-building constraint, then this voxel will be considered next. By construction, this makes it so that within each node of G_C, the build order does not violate the no-tight-building constraint. Together with Lemma 2.1, this makes it such that the whole order adheres to the constraint.

For a configuration consisting of n voxels, calculating the build order can be done in $O(n)$ time. During the first step, each edge and each voxel will only be considered once for the breath-first search. Since each voxel has a constant amount of neighbors, there are at most $O(n)$ edges in the graph and the whole breadth-first search runs in $O(n)$ time. Then, inside each component, we again run a breadth-first search, which takes $O(v)$, for a vertex of G_C consisting of v voxels. Because every voxels occurs in exactly one of these vertices, the total running time of the construction is $O(n)$.

Whenever a robot starts working on a node of G_C, they build all of the voxels in that node before moving on to another node. Nodes are built in order of the dependency graph. Whenever a robot is finished with building all voxels in a specific node, it moves on to another node for which all predecessors have been completely built. Whenever a robot r needs to be assigned to a new node, but there are no free nodes with all predecessors built, r will be asigned to the node with the highest remaining workload.

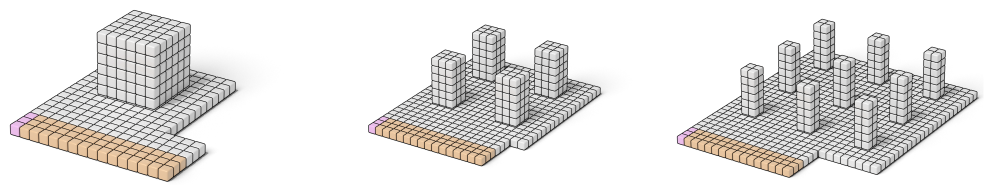


Figure 5 The three configurations. The robots start on top of the orange voxels. When a robot stands on top of the purple voxels representing the depot, it can obtain a voxel in its backpack. The base layer is present at the start of construction.

3 Evaluation

To evaluate our technique, we compare this build order with a lexicographical sorted order, which sorts the voxels in ascending order on z-, then y-, then x-coordinate. We compare these two approaches on several configurations, using different numbers of teams. Because we expect that our approach increases the amount of parallelism, we test the approaches on configurations that allow for a significant amount of parallelism.

We pick three configurations, with 216 voxels each, see Figure 5. We chose configurations that have different properties and different amounts of inherent parallelism. Configuration 1 is a $6 \times 6 \times 6$ block, configuration 2 consists of four pillars of 3×3 voxels and 6 voxels high, and lastly, configuration 3 has nine pillars of 2×2 voxels and 6 voxels high. The pillars are at least six spaces apart, such that robots walking on one pillar do not interfere with those on another pillar. We use Multi-label A* (MLA*) to calculate paths for the robots [14]. For a detailed description of the path planning approach see [15].

For each experiment, we calculate a baseline as follows. We take the number of steps it takes a single team to construct the lexicographical sorted order, and divide this by the number of teams. This gives an estimated lower bound on the total number of steps needed to build a structure if none of the teams would interfere with each other. Note that this is not a true lower bound on the time necessary for construction. After all, sometimes robot interference is unavoidable, and even for a single team, the running time is dependent on the build order, since placed voxels might interfere with future paths. However, when comparing the different construction order approaches to this baseline, it should give an impression of how much interference happens.

The results for Configuration 1 are visible in Figure 6. Our order performs better than the sorted build order. The advantage increases with the number of teams, until there are six teams, at which point is stays the same, with our approach taking approximately 78% of the time that the sorted order takes. Starting from around seven teams, adding more teams seems to not decrease the build time. This can be explained by the fact that the block is $6 \times 6 \times 6$ voxels. Adding more teams of robots makes them only interfere more. Therefore, the robots have to wait for each other to finish instead of working at the same time. Moreover, the effect of adding more teams decreases in general, as can be seen from the baseline. The difference between the actual results and the baseline represents the overhead caused by the interference between the robots. The observed build time for both the sorted order as well as our order does not decrease anymore, but the baseline does. This indicates that the more robots there are, the more they interfere.

A similar result can be seen for Configuration 2, visible in Figure 7. Here, the difference between our order and the sorted order increases until ten teams, then stays stable with our order taking approximately 76% of the time the sorted order takes.

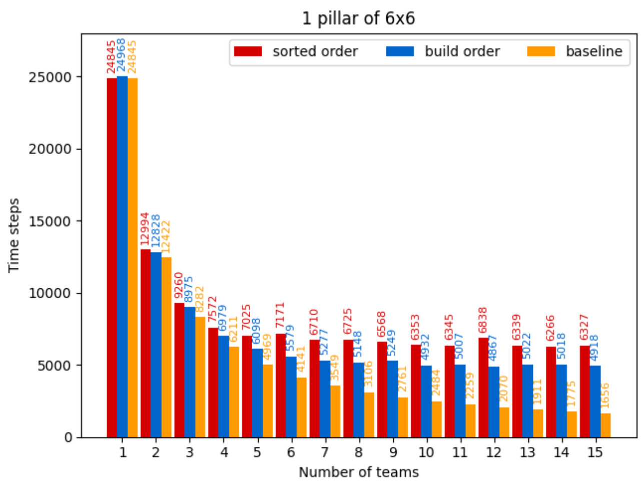


Figure 6 The timesteps it takes for different numbers of teams to finish building configuration 1.

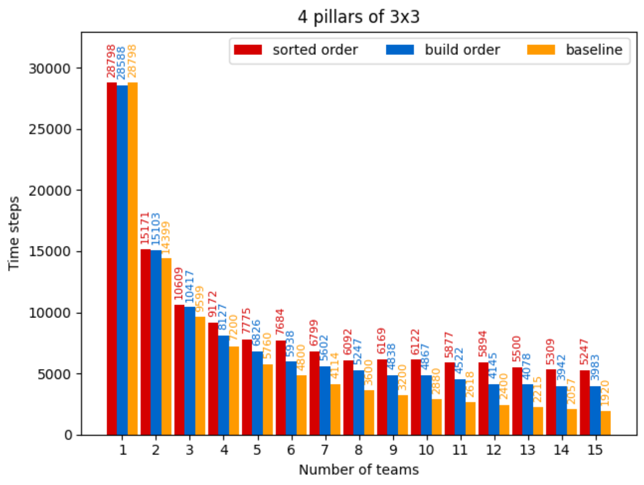


Figure 7 The timesteps it takes for different numbers of teams to finish building configuration 2.

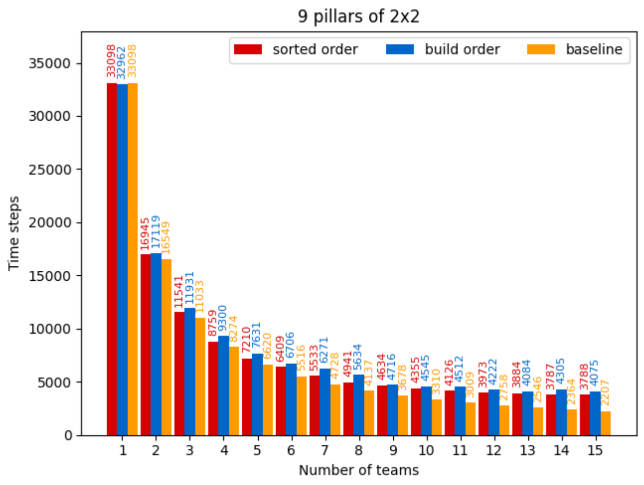


Figure 8 The timesteps it takes for different numbers of teams to finish building configuration 3.

Lastly, for Configuration 3, the build times for our order and the sorted order are approximately equal, with our order being slightly slower. This can be explained by the size of the pillars. Since the pillars are smaller, the amount of robots that get sent to each pillar in the lexicographical sorted order is smaller too and stays constant throughout the construction. This leads to less interference using this order. On the other hand, our build order divides the robots evenly over the pillars. This inevitably leads to some pillars being finished earlier than others. Eventually, once some pillars are done, our algorithm assigns available robots to the remaining pillars, leading to interference. This is a natural example of a Configuration with an inherent bottleneck. In this case, the bottleneck consists of narrow substructures with a rather low optimal number of robot teams. Such substructures must be constructed with care and we must try to not assign too many robots to it.

4 Conclusion

We presented an algorithm for dividing a structure over teams of robots to be built. We tested our algorithm on three different Configurations, each with different inherent amounts of parallelism. Our build order is better when the Configuration contain no inherent bottlenecks. This is due to the fact that having more robots work together on a narrow walkway does not necessarily lead to faster build times, and instead might even slow it down.

A simple solution to this problem might be to try to detect these bottlenecks and only assign a limited number of robots to them. Another approach is to change up the team configuration. The armadas architecture uses teams of three robots each. However, it could be beneficial to have SOLL-E's swap between placing voxels and carrying voxels to the construction site, instead of having dedicated crane and cargo robots.

5 Acknowledgements

This work was supported by the NASA Game Changing Development (GCD) Program, Space Technology Mission Directorate.

References

1 J. Romanishin and D. Rus, "Dynamic and repeatable modular assembly: How kinematic couplings and proprioceptive actuators simplify robotic assembly," in *International Symposium on Experimental Robotics.* Springer, 2023, pp. 308–318.

2 C. Wagner, N. Dhanaraj, T. Rizzo, J. Contreras, H. Liang, G. Lewin, and C. Pinciroli, "Smac: Symbiotic multi-agent construction," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3200–3207, 2021.

3 B. Jenett and K. Cheung, "BILL-E: Robotic platform for locomotion and manipulation of lightweight space structures," in *25th AIAA/AHS Adaptive Structures Conference*, 2017, p. 1876.

4 G. Sartoretti, Y. Wu, W. Paivine, T. S. Kumar, S. Koenig, and H. Choset, "Distributed reinforcement learning for multi-robot decentralized collective construction," in *14th International Symposium on Distributed Autonomous Robotic Systems.* Springer, 2019, pp. 35–49.

5 E. Lam, P. J. Stuckey, S. Koenig, and T. S. Kumar, "Exact approaches to the multi-agent collective construction problem," in *26th International Conference on Principles and Practice of Constraint Programming.* Springer, 2020, pp. 743–758.

6 A. K. Srinivasan, S. Singh, G. Gutow, H. Choset, and B. Vundurthy, “Multi-agent collective construction using 3d decomposition,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 9963–9969.

7 S. Singh, G. Gutow, A. K. Srinivasan, B. Vundurthy, and H. Choset, “Hierarchical propositional logic planning for multi-agent collective construction,” in *Construction Robotics Workshop*, 2023.

8 J. Brunner, K. Cheung, E. Demaine, J. Diomidova, H. D. Christine Gregg, and I. Kostitsyna., “Reconfiguration algorithms for cubic modular robots with realistic movement constraints,” in *Proc. 19th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, 2024, pp. 34:1–34:18.

9 I.-W. Park, D. Catanoso, O. Formoso, C. Gregg, M. Ochalek, T. Olatunde, F. Sebastianelli, P. Spino, E. Taylor, G. Trinh *et al.*, “SOLL-E: A module transport and placement robot for autonomous assembly of discrete lattice structures,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 10 736–10 741.

10 O. Formoso, G. Trinh, D. Catanoso, I.-W. Park, C. Gregg, and K. Cheung, “MMIC-i: A robotic platform for assembly integration and internal locomotion through mechanical meta-material structures,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7303–7309.

11 C. E. Gregg, D. Catanoso, O. I. B. Formoso, I. Kostitsyna, M. E. Ochalek, T. J. Olatunde, I. W. Park, F. M. Sebastianelli, E. M. Taylor, G. T. Trinh, and K. C. Cheung, “Ultralight, strong, and self-reprogrammable mechanical metamaterials,” *Science Robotics*, vol. 9, no. 86, p. eadi2746, 2024.

12 B. Bernus, G. Trinh, C. Gregg, O. Formoso, and K. Cheung, “Robotic specialization in autonomous robotic structural assembly,” in *2020 IEEE Aerospace Conference*. IEEE, 2020, pp. 1–10.

13 C. Sung, J. Bern, J. Romanishin, and D. Rus, “Reconfiguration planning for pivoting cube modular robots,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1933–1940.

14 F. Grenouilleau, W.-J. Van Hoeve, and J. N. Hooker, “A multi-label A* algorithm for multi-agent pathfinding,” in *Proc. International Conference on Automated Planning and Scheduling*, vol. 29, 2019, pp. 181–185.

15 I. Kostitsyna, K. Cheung, and J. Gloyd, “Multi-agent collective construction of general modular structures,” *to appear at ICRA 2025*, 2025.

Saturated Drawings of Geometric Thickness k *

Patricia Bachmann[1], Anna Brötzner[†2], Miriam Goetze[‡3], Philipp Kindermann[4], Matthias Pfretzschner[§1], and Soeren Terziadis[¶5]

1 **Chair of Theoretical Computer Science, University of Passau, Germany,** `bachmanp@fim.uni-passau.de`, `pfretzschner@fim.uni-passau.de`
2 **Department of Computer Science and Media Technology, Malmö University, Sweden,** `anna.brotzner@mau.se`
3 **Institute of Theoretical Informatics, Karlsruhe Institute of Technology, Germany,** `miriam.goetze@kit.edu`
4 **Algorithms Group, Trier University, Germany,** `kindermann@uni-trier.de`
5 **Algorithms Group, TU Eindhoven, The Netherlands,** `s.d.terziadis@tue.nl`

Abstract

We investigate saturated geometric drawings of graphs with geometric thickness k, where no edge can be added without increasing k. We establish lower and upper bounds on the number of edges in such drawings if the vertices lie in convex position. We also study the more restricted version where edges are precolored, and for $k = 2$ the case for vertices in non-convex position.

Related Version : `Full version available on arXiv.` [3]

1 Introduction

The *geometric thickness* $\bar{\theta}(G)$ of a graph G is the minimum number k such that there exists a straight-line drawing Γ of G and a k-edge-coloring $\varphi\colon E(G) \to \{1, \ldots, k\}$ that has no monochromatic crossings, see Fig. 1 for an example. We also write $E_i \subseteq E(G)$ to denote all edges of color i. We call Γ a *Θ^k-drawing* (with thickness k). When the coloring φ of Γ is given, we say that Γ is *precolored* (we always assume that in a given coloring, there are no monochromatic crossings). If the vertices in Γ are in convex position, Γ is *convex*. Connecting all vertices of the outer face of a Θ^k-drawing with edges along the convex hull

* This research was initiated at GGWeek 2024 in Trier. We would like to thank the organizers and participants of the workshop for the friendly and supportive environment and the fruitful discussions.
† supported by grant 2021-03810 from the Swedish Research Council (Vetenskapsrådet).
‡ funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 520723789
§ funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 541433306
¶ funded by the NWO Gravitation project NETWORKS under grant no. 024.002.003.

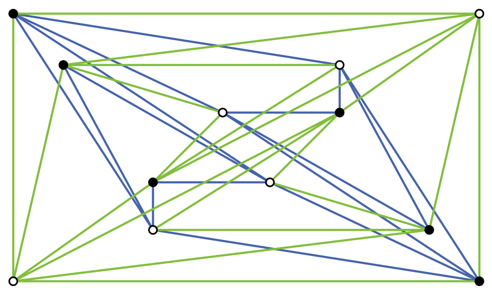


Figure 1 Taken from [8, Fig. 2]. A drawing of the non-planar graph $K_{6,6}$ witnessing $\bar{\theta}(K_{6,6}) = 2$.

yields a cycle. We call this cycle the *outer cycle* of Γ, and an edge e on this cycle an *outer edge* of Γ. All other edges are *inner edges*. Note that not all edges of the outer cycle are necessarily contained in Γ.

We call a Θ^k-drawing Γ of a graph G *saturated* if there are no two vertices $u, v \in V(G)$ with $uv \notin E(G)$ such that the drawing $\Gamma' = \Gamma + uv$ with the edge uv drawn as a straight line is also a Θ^k-drawing. If Γ is precolored, we require that Γ' uses the same coloring, i.e., only the color of uv may vary. If Γ has the minimum (maximum) number of edges among all saturated Θ^k-drawing on the same number of vertices, it is *min-saturated* (*max-saturated*). We assume that vertices lie in general position, i.e., there are no three vertices on a line.

Max- and min-saturation have similarly been defined for graph classes instead of drawings. There is a rich history of results analyzing max-saturated graphs (Turán type results, following seminal work by Turán [14]). It is widely known that max-saturated planar graphs contain $3n - 6$ edges, and bounds have been proven for several beyond planar graph classes. For example, 1-planar and 2-planar max-saturated graphs have $4n - 8$ and $5n - 10$ edges, respectively [13], while general k-planar max-saturated graphs are only known to have at most $3.81\sqrt{k}n$ edges [1]. Similar results have recently been shown for min-k-planar graphs [5].

The study of min-saturated graphs builds on the work of Erdős, Hajnal, and Moon [10], who characterize min-saturated K_k-free graphs. A survey [7] with a recent second edition provides an overview of results in this direction. While min-saturated planar graphs also contain $3n - 6$ edges, min-saturated 1-planar graphs only have at most $\frac{45}{17}n + O(1)$ edges. Chaplick et al. [6] recently investigated the number of edges in min-saturated (not necessarily straight-line) k-planar drawings under a variety of drawing restrictions.

Graphs of geometric thickness k form a relevant beyond-planar graph class. The concept was first introduced by Kainen [12] (who used the term linear thickness) and later investigated by Dillencourt, Eppstein, and Hirschberg [8], who considered the geometric thickness of complete and complete bipartite graphs. Checking whether a graph has geometric thickness at most k has been shown to be NP-hard [9] even for $k \leq 2$ and for multigraphs it is $\exists\mathbb{R}$-complete [11] for $k \leq 30$. In fact, a graph G has stack number at most k if and only if it admits a convex Θ^k-drawing. That is, in the convex setting, we investigate the min-saturation of graphs with stack number at most k.

We provide upper and lower bounds on the number of edges in min-saturated Θ^k-drawings in the precolored and non-precolored, as well as in the convex and non-convex setting. After presenting upper bounds for convex precolored and non-precolored Θ^k-drawings in Section 2.1, we give lower bounds for Θ^3-drawings (applying to the precolored and non-precolored setting) in Section 2.2. In Section 3, we present a lower bound for non-convex non-precolored Θ^2-drawings and conclude in Section 4. Results marked with ($\star$) are proven in the full version [3].

2 Convex Drawings

Each color class of a convex Θ^k-drawing induces an outerplane graph H. For $\ell \geq 3$, we call an outerplane graph H an *inner ℓ-angulation* if every inner face has size ℓ and the outer face is a simple cycle. Inner 3-angulations and inner 4-angulations are called *inner triangulations* and *inner quadrangulations*, respectively. Double-counting the edge-face-incidences shows that every inner ℓ-angulation with n vertices and f faces contains $\frac{1}{2}(n + \ell(f - 1))$ edges. Now Euler's formula implies:

▶ **Observation 2.1.** *For $\ell \geq 3$, every inner ℓ-angulation of a graph on $n \geq \ell$ vertices contains $\frac{n-\ell}{\ell-2}$ inner edges.*

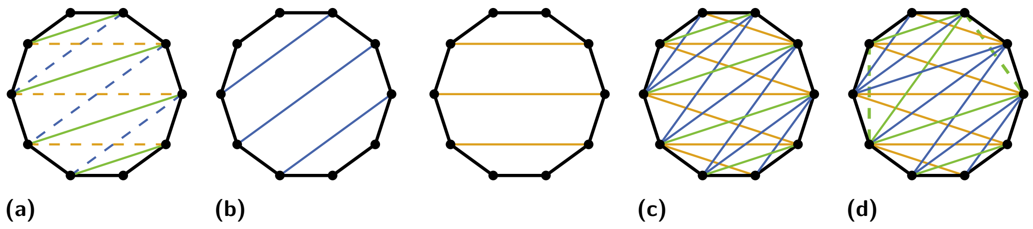


Figure 2 (a) A nice matching M (green) and diagonals that could be added to a color class containing M (dashed). (b) The two nice matchings $L(M)$ (left) and $R(M)$ (right). (c) A precolored Θ^3-zigzag Γ. (d) Recoloring yields a Θ^3-drawing which contains Γ and two more edges (dashed).

2.1 Bounds for Saturated Convex Θ^k-Drawings

Note that each color class of a convex Θ^k-drawing of an n-vertex graph is a subgraph of some inner triangulation. That is, we can cover the edges of the Θ^k-drawing with k inner triangulations, any two of which only share the outer cycle. Now, Observation 2.1 yields the following upper bound on the number of edges.

▶ **Proposition 2.2** ([4, Theorem 3.3]). *Every convex Θ^k-drawing of a graph G on $n \geq 3$ vertices contains at most $n + k(n-3)$ edges.*

We now construct a precolored saturated drawing with a smaller number of edges than implied by Proposition 2.2, thereby obtaining a smaller upper bound for precolored min-saturated drawings. We say that some diagonals M of a convex Θ^k-drawing of a graph on n vertices form a *nice matching* if these diagonals together with the outer cycle form an outerplane graph H whose dual is a path (ignoring the outer face) and where the faces corresponding to the beginning and end of the path are faces of size 3 or 4, and all other faces have size 4, see Fig. 2a. If all these diagonals belong to the same color class E_i, then E_i can only be extended by adding missing diagonals within the faces of H. The missing diagonals may again be decomposed into two nice matchings, which we call the *left* and *right tilt* of M, denoted by $L(M)$ and $R(M)$, respectively, see Fig. 2b. In particular, we have $R(L(M)) = M$ and $L(R(M)) = M$. We can now construct a saturated Θ^k-drawing Γ of a graph G on n vertices with an edge-coloring $\varphi\colon E(G) \to \{1, \dots, k\}$ such that the following holds (see Fig. 2c for an example):

- The outer cycle is part of Γ
- The inner edges of E_1 correspond to two nice matchings M_1 and $L(M_1)$
- The inner edges of E_i form a nice matching $M_i = R(M_{i-1})$, for $i = 2, \dots, k-1$
- The inner edges of E_k correspond to two nice matchings $M_k = R(M_{k-1})$ and $R(M_k)$

We call the obtained precolored drawing a *precolored Θ^k-zigzag* Γ on n vertices. Here, no E_i can be extended as all the edges that could be added to E_i are part of some E_j with $j \neq i$. That is, the Θ^k-zigzag Γ is a precolored saturated drawing. For $k \leq \frac{n}{2}$ we have $E_i \cap E_j = \emptyset$, i.e., disjoint edge sets and Γ is well-defined.

▶ **Proposition 2.3.** *Every min-saturated convex precolored Θ^k-drawing of a graph G on $n \geq 5$ vertices (with $k \leq \frac{n}{2}$) contains at most $\frac{1}{2}(k+4)(n-2)$ edges.*

Proof. Consider the precolored Θ^k-zigzag on n vertices. By Observation 2.1, E_1 and E_k contain at most $n-3$ inner edges respectively. Every nice matching together with the outer cycle is an inner quadrangulation except for at most two faces of complexity 3. A similar argument as in Observation 2.1 shows that every nice matching contains at most $\frac{1}{2}(n-2)$ edges. Summing up the number of edges of the outer cycle (n edges), the inner edges of E_1 and E_k $(2(n-3))$, and the edges of the $k-2$ nice matchings E_i $(\frac{1}{2}(n-2)$ each) yields the desired bound. ◀

Yet, this upper bound does not yield an upper bound for non-precolored drawings. Indeed, a Θ-zigzag (without the edge-coloring) is not necessarily saturated, cf. Fig. 2c and Fig. 2d.

Recall that every color class of a Θ^k-drawing together with the outer cycle forms an outerplane graph. For max-saturated precolored drawings, the inner faces of these outerplane graphs cannot have arbitrarily large size:

▶ **Lemma 2.4** ($\star$). *If Γ is a saturated precolored convex Θ^k-drawing, then each color class of Γ together with the outer cycle forms an outerplane drawing where each inner face has size at most $2k-1$.*

Thus, by Lemma 2.4, we can cover the edges of a saturated precolored Θ^k-drawing with k outerplane graphs, each of which contains an inner $(2k-1)$-angulation that contains the edges of the outer cycle. An application of Observation 2.1 yields the following.

▶ **Theorem 2.5.** *Every min-saturated convex precolored Θ^k-drawing of a graph on $n \geq 2k-1$ vertices contains at least $\frac{k(n-2k+1)}{2k-3}+n$ edges.*

Note that, since the upper bounds implied by Proposition 2.2 and Proposition 2.3 and the lower bound of Theorem 2.5 coincide for Θ^2-drawings, we obtain the following.

▶ **Corollary 2.6.** *Every saturated (precolored) convex Θ^2-drawing Γ of a graph G on $n \geq 3$ vertices contains exactly $3n-6$ edges.*

Note that the number of edges of saturated convex Θ^2-drawings only depends on the number of vertices, that is, min- and max-saturated Θ^2-drawings coincide. This is different from other results related to saturation problems. For example, there are saturated 2-planar drawings of graphs on n vertices that contain only $1.33n$ edges [2], while the maximum number of edges in saturated 2-planar drawings is $5n$ [13]. In particular, Corollary 2.6 shows that even if we fix the edge-coloring that certifies geometric thickness k (when considering precolored drawings), the number of edges in every saturated convex Θ^2-drawing is $3n-6$.

2.2 Edge-Density of Saturated Convex Θ^3-Drawings

With $k=2$ being covered by the general bounds of the previous section, we now turn to $k=3$. In the case of Θ^3-drawings, we can strengthen the result of Lemma 2.4 as follows.

▶ **Lemma 2.7.** *If Γ is a saturated precolored convex Θ^3-drawing, then each color class of Γ and the outer cycle forms an outerplane drawing Γ' where all inner faces have size at most 4.*

Proof. Let Γ be a saturated precolored convex Θ^3-drawing with colors ***blue***, ***green*** and ***red*** and let Γ' be the outerplane drawing induced by the **red** edges and the outer cycle. Suppose some inner face f of Γ' contains at least five vertices $v_1, \dots, v_5$. Each diagonal v_iv_j with $i \neq j$ is colored in **blue** or **green**. Note that the conflict graph H whose vertices are the diagonals v_iv_j and whose edges are pairs of crossing diagonals is a 5-cycle, see Fig. 3 for an example. Yet, the 2-edge-coloring of the diagonals induces a proper 2-vertex coloring of the 5-cycle H, a contradiction. Thus, every inner face has size at most 4. ◀

Thus, every color class of a saturated convex precolored Θ^3-drawing together with the outer cycle forms an outerplane drawing that contains an inner quadrangulation. Now Observation 2.1 yields the following.

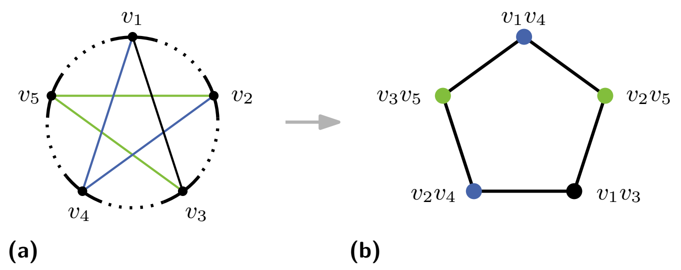


Figure 3 (a) Five vertices on a face of size at least 5 in Γ_r and their diagonals in Γ. The black edge is not present ins Γ. (b) The corresponding vertex-coloring of the conflict graph H.

▶ **Theorem 2.8.** *Every saturated precolored convex Θ^3-drawing Γ of a graph G on $n \geq 3$ vertices contains at least $\frac{5}{2}n - 6$ edges.*

If a Θ^3-drawing is saturated for every 3-edge-coloring (with no monochromatic crossings), the lower bound on the number of edges can be improved.

▶ **Theorem 2.9.** *Every saturated convex Θ^3-drawing of a graph G on $n \geq 3$ vertices contains at least $\frac{7}{2}n - 8$ edges.*

Proof. Let Γ be a saturated convex Θ^3-drawing of G. That is, no edge can be added to Γ, independent of the 3-edge-coloring we consider. We call the three colors of a corresponding edge-coloring of Γ ***blue***, ***green***, and ***red***. Greedily adding missing diagonals in **blue** or **green**, we may assume that the union of the **blue** edges, the **green** edges, and the outer cycle is a saturated Θ^2-drawing Γ'. In fact, as Γ is saturated, we only recolor some **red** edges in the process. By Corollary 2.6, the subdrawing Γ' contains $3n - 6$ edges.

It remains to show that there are at least $\frac{n}{2} - 2$ **red** inner edges. As Γ is saturated (for every coloring), the **red** edges together with the outer cycle form a drawing that contains an inner quadrangulation (cf. Lemma 2.7). Thus, by Observation 2.1, there are at least $\frac{n}{2} - 2$ **red** inner edges. ◀

3 Moving towards non-convexity in the free setting for $k = 2$

In this section, we consider the more general case where the vertices of G are not necessarily in convex position. We show that, for $k = 2$, the lower bound from Section 2 (cf. Corollary 2.6) extends to the general case, i.e., we prove the following theorem.

▶ **Theorem 3.1** ($\star$). *Every saturated Θ^2-drawing of a graph G on $n \geq 3$ vertices contains at least $3n - 6$ edges.*

Proof Sketch. Let Γ be a Θ^2-drawing of G. We show that we can always add additional edges to Γ without increasing its thickness to more than 2 if Γ contains fewer than $3n - 6$ edges. We assume that the edges of Γ are colored **blue** and **red** according to an arbitrary certificate of its thickness. Let n' be the number of vertices that lie on the outer cycle. Adding missing edges and recoloring some of the **red** edges in **blue**, we greedily turn the **blue** edges into a plane graph where each inner face is a triangle and the outer face is bounded by the outer cycle. That is, we may assume that there are $3n - 6 - (n' - 3)$ **blue** edges by Observation 2.1. In order to obtain the desired lower bound of $3n - 6$ edges, we thus need to obtain at least $n' - 3$ **red** edges overall.

Consider an ordering $e_1, \ldots, e_t$ of the **red** edges. We iteratively extend each edge e_i to a line segment as follows. We say that two line segments *cross* if there exists a point p that

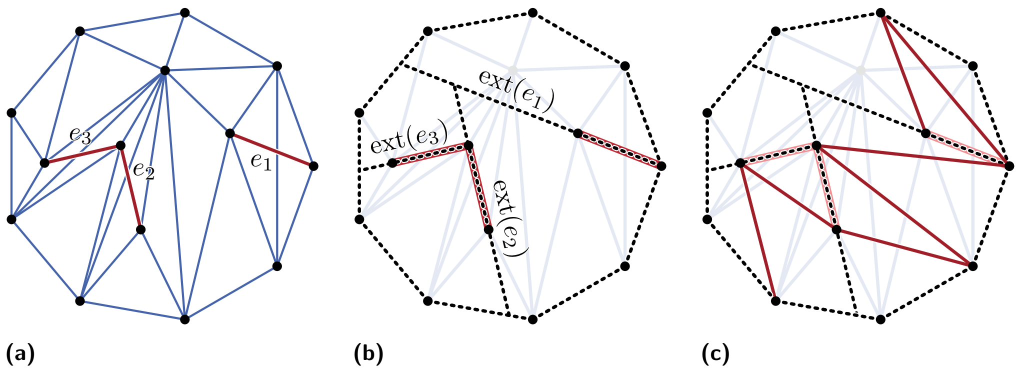


Figure 4 (a) A Θ^2-drawing Γ where the blue edges form an inner triangulation. (b) The corresponding drawing Λ. (c) Triangulating each inner cell of Λ yields seven additional red edges. Note that, while the red edges do not form an inner triangulation of the whole graph, we now have at least $3n - 6$ edges overall as desired.

lies in the interior of both segments (i.e., p is not an endpoint of either segment). Let ℓ_i be the supporting line of e_i. We define the *edge extension of e_i*, denoted $\mathrm{ext}(e_i)$, as the segment of ℓ_i of maximum length that contains e_i and does not cross any e_j with $j \neq i$, the outer cycle, or any extension $\mathrm{ext}(e_j)$ with $j < i$; see Figure 4. Note that, if two edge extensions $\mathrm{ext}(e_i)$ and $\mathrm{ext}(e_j)$ share a point p, then p is an endpoint of at least one of them. We say that $\mathrm{ext}(e_i)$ and $\mathrm{ext}(e_j)$ *touch* in the point p. If an extension $\mathrm{ext}(e_j)$ touches an extension $\mathrm{ext}(e_i)$ in an inner point of $\mathrm{ext}(e_i)$, we say $\mathrm{ext}(e_j)$ splits $\mathrm{ext}(e_i)$ into *segments*. Observe that every vertex that does not lie on the outer cycle lies in the interior of exactly one edge extension, but may be the endpoint of other edge extensions.

We denote by Λ the drawing induced by the outer cycle together with all edge extensions. The drawing Λ splits the plane into regions that we call *cells*. We denote by $C(\Lambda)$ the set of inner cells of the drawing Λ. The *boundary* of a cell c corresponds to all segments and vertices incident to c. We let $\|c\|$ denote the number of vertices on the boundary of c.

Each cell $c \in C(\Lambda)$ is convex. Using a double counting argument for the vertex-cell incidences, we can show that the sum of these values over all cells of Λ plus the initial number of **red** inner edges in Γ adds up to at least $n' - 3$, the desired number of **red** edges. ◀

4 Conclusion

We investigated saturated geometric drawings of graphs on n vertices with geometric thickness k. We provided upper and lower bounds on the number of edges in such drawings, and took a closer look at drawings of thickness $k = 2$ and $k = 3$. Several questions remain open, e.g., tight bounds for the convex case, and lower and upper bounds for min-saturated drawings with n' vertices on the convex hull.

References

1 Eyal Ackerman. On topological graphs with at most four crossings per edge. *Computational Geometry*, 85, 2019. doi:10.1016/J.COMGEO.2019.101574.

2 Christopher Auer, Franz J. Brandenburg, Andreas Gleißner, and Kathrin Hanauer. On sparse maximal 2-planar graphs. *Proceedings of the 20th International Symposium on Graph Drawing (GD'12)*, pages 555–556, 2013. doi:10.1007/978-3-642-36763-2_50.

3 Patricia Bachmann, Anna Brötzner, Miriam Goetze, Philipp Kindermann, Matthias Pfretzschner, and Soeren Terziadis. Saturated drawings of geometric thickness k. arXiv preprint, 2025. URL: `http://arxiv.org/abs/2503.03577`.

4 Frank Bernhart and Paul C. Kainen. The book thickness of a graph. *Journal of Combinatorial Theory, Series B*, 27(3):320–331, 1979. `doi:10.1016/0095-8956(79)90021-2`.

5 Carla Binucci, Aaron Büngener, Giuseppe Di Battista, Walter Didimo, Vida Dujmovic, Seok-Hee Hong, Michael Kaufmann, Giuseppe Liotta, Pat Morin, and Alessandra Tappini. Min-k-planar drawings of graphs. *Journal of Graph Algorithms and Applications*, 28(2):1–35, 2024. `doi:10.7155/JGAA.V28I2.2925`.

6 Steven Chaplick, Fabian Klute, Irene Parada, Jonathan Rollin, and Torsten Ueckerdt. Edge-minimum saturated k-planar drawings. *Journal of Graph Theory*, 106(4):741–762, 2024. `doi:10.1002/jgt.23097`.

7 Bryan L. Currie, Jill R. Faudree, Ralph J. Faudree, and John R. Schmitt. A survey of minimum saturated graphs. *The Electronic Journal of Combinatorics*, DS19:1–98, 2011. `doi:10.37236/41`.

8 Michael B. Dillencourt, David Eppstein, and Daniel S. Hirschberg. Geometric thickness of complete graphs. *Journal of Graph Algorithms and Applications*, 4(3):5–17, 2000. `doi:10.7155/JGAA.00023`.

9 Stephane Durocher, Ellen Gethner, and Debajyoti Mondal. Thickness and colorability of geometric graphs. *Computational Geometry*, 56:1–18, 2016. `doi:10.1016/j.comgeo.2016.03.003`.

10 Paul Erdős, András Hajnal, and John W. Moon. A problem in graph theory. *The American Mathematical Monthly*, 71(10):1107–1110, 1964. `doi:10.2307/2311408`.

11 Henry Förster, Philipp Kindermann, Tillmann Miltzow, Irene Parada, Soeren Terziadis, and Birgit Vogtenhuber. Geometric thickness of multigraphs is $\exists\mathbb{R}$-complete. In José A. Soto and Andreas Wiese, editors, *Proceedings of the 16th Latin American Symposium (LATIN'24)*, volume 14578 of *Lecture Notes in Computer Science*, pages 336–349. Springer, 2024. `doi:10.1007/978-3-031-55598-5_22`.

12 Paul C. Kainen. Thickness and coarseness of graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 39(1):88–95, 1973. `doi:10.1007/BF02992822`.

13 János Pach and Géza Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17(3):427–439, 1997. `doi:10.1007/BF01215922`.

14 Pál Turán. On an extremal problem in graph theory. *Matematikai és Fizikai Lapok*, 48:436–452, 1941. URL: `https://zbmath.org/0026.26903`.

Unit Edge-Length Rectilinear Drawings with Crossings and Rectangular Faces*

Patrizio Angelini[1], Carla Binucci[2], Giuseppe Di Battista[3], Emilio Di Giacomo[2], Walter Didimo[2], Fabrizio Grosso[4], Giacomo Ortali[2], and Ioannis G. Tollis[5]

1 **John Cabot University** pangelini@johncabot.edu
2 **University of Perugia**
{carla.binucci,emilio.digiacomo,walter.didimo,giacomo.ortali}@unipg.it
3 **University of Roma Tre** giuseppe.dibattista@uniroma3.it
4 **CeDiPa - University of Perugia** fabrizio.grosso@unipg.it
5 **University of Crete** tollis@csd.uoc.gr

Abstract

Unit Edge-length Rectilinear drawings with Rectangular Faces (UER-RF drawings) are drawings of graphs where all edges have unit length and are represented as either horizontal or vertical segments, and where all faces are rectangles. We study UER-RF drawings that can have edge crossings. We consider crossings as dummy vertices and apply the unit edge-length convention to the edge segments connecting any two (real or dummy) vertices. We present several efficient algorithms to recognize graphs that admit UER-RF drawings and to construct such drawings if they exist. We first consider drawings such that removing the external face yields a collection of paths and cycles; we then study the general case. We consider both the general unconstrained setting and the setting where either the external boundary of the drawing or the rotation system of the graph are given in input.

1 Introduction

Unit Edge-length Rectilinear drawings with Rectangular Faces (*UER-RF drawings*) are drawings of graphs where all edges have unit length and are represented as either horizontal or vertical segments (rectilinear drawings), and where all faces are convex (i.e., rectangles). We study non-planar UER-RF drawings, considering each crossing as a "dummy" vertex and applying the unit edge-length constraint to the edge segments connecting any two vertices, either real or dummy. Examples of UER-RF drawings are presented in Figure 1.

Recognizing graphs that admit planar straight-line drawings with all edges of the same length is NP-hard [5, 6, 7] and, even stronger, $\exists\mathbb{R}$-complete [1, 12]. Testing whether a graph admits a rectilinear planar drawing is NP-hard [8, 9], whereas it is polynomial-time solvable if the planar embedding is given as part of the input, even if the faces are required to be rectangles [10, 11, 13].

Alegría et al. [2, 3] show that recognizing planar graphs admitting planar UER-RF drawings is feasible in polynomial time. We study the recognition and the construction of non-planar UER-RF drawings when the rotation system and/or the external face are or are not part of the input. In our problem, the assignment of angles around the vertices plays an important role. Requiring rectangular faces enforces 180° angles at degree-2 vertices, and the challenge

* This research was supported, in part, by MUR of Italy (PRIN Project no. 2022ME9Z78 – NextGRAAL and PRIN Project no. 2022TS4Y3N – EXPAND). The sixth author was supported by Ce.Di.Pa. - PNC Programma unitario di interventi per le aree del terremoto del 2009-2016 - Linea di intervento 1 sub-misura B4 - "Centri di ricerca per l'innovazione" CUP J37G22000140001.

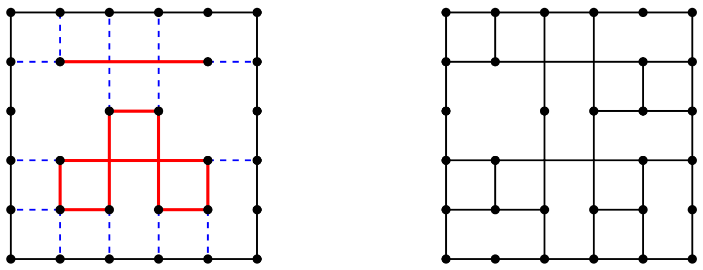


Figure 1 (Left) A UER-RF drawing whose internal vertices induce a collection of paths and cycles; (Right) A UER-RF drawing in the general setting.

arises with degree-3 vertices. In view of this, we first present an efficient solution for a restricted scenario and then provide an FPT algorithm for the general case, parameterized by the number of degree-3 vertices. Let G be an n-vertex graph; we present the following results:

– Polynomial-time testing and construction algorithms for UER-RF drawings in the restricted scenario where removing the external face yields a collection of paths and cycles (Section 3). The complexity of these algorithms ranges from $O(n^2)$ to $O(n^4)$, depending on the specific constraints on the external face and on the rotation system.

– An $O(3^k n^{4.5})$-time testing and construction algorithm for general UER-RF drawings, where k is the number of degree-3 vertices (Section 4).

An extended abstract of this paper can be found in [4]

2 Basic Definitions

Let $V(G)$ and $E(G)$ be the vertex and edge sets of a graph G, respectively. For a vertex $v \in V(G)$, let $N(v)$ be the set of *neighbors of* v and let $\deg_G(v) = |N(v)|$ be the *degree* of v. A graph G is a *k-graph* ($k > 0$) if $\deg_G(v) \leq k$ for all $v \in V(G)$.

A *vertex of* a UER-RF drawing Γ of G is called either a *real-vertex*, if it corresponds to a vertex of G, or a *crossing-vertex*, if it corresponds to a point where two edges of G cross. An *edge of* Γ is a portion of an edge of G delimited by two vertices of Γ, with no other vertices of Γ in its interior. It coincides with an edge e of G if e has no crossings in Γ. Drawing Γ divides the plane into connected regions, called *faces*. The *boundary* of each face f is the circular sequence of vertices (either real- or crossing-vertices) and edges of Γ that delimit f. The unique infinite region is the *external face* of Γ; the other faces are the *internal faces* of Γ.

A *rotation system* $\mathcal{R}(G)$ *of* G specifies the clockwise order of the edges in $E(v)$, for each vertex $v \in V(G)$. A (UER-RF) drawing Γ of G determines a rotation system for G; in addition, Γ determines the clockwise order of the edges incident to each crossing-vertex. For a given rotation system $\mathcal{R}(G)$, we say that Γ *preserves* $\mathcal{R}(G)$ if the rotation system determined by Γ coincides with $\mathcal{R}(G)$.

▶ Property 1. Let Γ be a UER-RF drawing of a graph G, let C be the external cycle of Γ, and let c_1, c_2, c_3, c_4 be the vertices of C at the corners of Γ. Then: (*i*) c_1, c_2, c_3, c_4 have degree 2 in G; (*ii*) each other vertex of C has degree at most 3 in G; (*iii*) the path in C between two consecutive corners has the same length as the path in C between the other two corners.

We will assume that the input graph G is biconnected and it is not a cycle, as biconnectivity is necessary for the existence of UER-RF drawings, and if G is a cycle the test is trivial.

3 Internal Paths and Cycles

In this section we consider the restricted scenario in which the input 4-graph G contains a cycle C such that $G \setminus C$ has vertex-degree at most two, i.e., it is a collection of paths and cycles (see, e.g., Figure 1(Left)). We call G an *inner-2-graph with respect to C*. We assume that C is given as part of the input.

▶ **Theorem 3.1.** *Let G be an inner-2-graph with respect to a given cycle C. There exists a polynomial-time algorithm that tests whether G admits a UER-RF drawing whose external cycle coincides with C. If such a drawing exists, the algorithm constructs one. If the four corners are prescribed, the algorithm takes $O(n^2)$ time, otherwise it takes $O(n^4)$ time. Furthermore, the algorithm can be adapted to preserve a given rotation system.*

Proof sketch. We assume that the four corners are given. This uniquely defines the rectangle that represents C in the drawing. If the four corners are not given, we guess $O(n^2)$ pairs of degree-2 vertices to be consecutive corners and infer the other two based on Condition (iii) of Property 1, thus obtaining $O(n^2)$ candidate rectangles.

Let Γ_C be one of the candidate rectangles computed in the previous step. We show how to place the vertices of $G \setminus C$ in the interior of Γ_C. Assume, w.l.o.g., that the bottom-left corner of Γ_C has coordinates $(0, 0)$. Let W and H be the maximum x and y coordinates of Γ_C, respectively. We traverse the points (i, j) $(1 \leq i \leq W - 1$ and $1 \leq j \leq H - 1)$ of the grid that lie internally to Γ_C from left to right and secondarily from top to bottom starting from the top-leftmost one, which has coordinates $(1, H - 1)$. We call *placed vertices* those vertices whose coordinates have already been decided (i.e., the vertices of C and those already placed at some internal point). When processing the point (i, j) we call *fixed vertices* the placed vertices with coordinates (i', j') such that either $i' < i$ or $i' = i$ and $j' > j$ (see Figure 2).

During the traversal we maintain two arrays of vertices Up and Left of size W and H, respectively; see Figure 2. Intuitively, the meaning of these arrays is as follows. When we process the point (i, j), let q be the rightmost fixed vertex such that $y(q) = j$; analogously, let r be the bottommost fixed vertex such that $x(r) = i$. If Left$[j]$ stores a vertex q', then q' is adjacent to q and $y(q')$ will be equal to j whereas $x(q')$ has not been decided yet; in other words, it is already decided that q' will be horizontally aligned with q (and thus it is candidate to occupy point (i, j)). If Left$[j]$ is null, then no neighbor of q will be assigned y-coordinate j; this implies that (i, j) cannot be occupied by a crossing. Analogously, if Up$[i]$ stores a vertex r', then it is already decided that r' will be vertically aligned with r; if Up$[i]$ is null, then no neighbor of r will be assigned x-coordinate i.

We initialize the array Left as follows. For $1 \leq j \leq H - 1$, let v_j be the vertex of the left side of Γ_C such that $y(v_j) = j$. If $\deg_G(v_j) = 2$, we set Left$[j] =$ null. If $\deg_G(v_j) = 3$, we set Left$[j] = u_j$, where u_j is the neighbor of v_j that is not on the left-side of Γ_C. The vertex u_j is either on the right side of Γ_C or it is an internal vertex of G. If $u_j \in C$ and $y(u_j) \neq y(v_j)$, we reject Γ_C. The Up array is initialized similarly, considering the vertices on the top side of Γ_C, and their neighbors that do not lie on the top side of Γ_C (if any).

We now explain how to process the generic point (i, j). We distinguish five cases depending on the values in the Left and Up arrays (see also Figure 2) **1.** Left$[j] =$ Up$[i]$ and they are both not null; **2.** Left$[j] \neq$ Up$[i]$ and they are both not null; **3.** Left$[j] \neq$ null and Up$[i] =$ null; **4.** Left$[j] =$ null and Up$[i] \neq$ null; and **5.** Left$[j] =$ Up$[i]$ and they are both null. In Case 1, let v be the vertex stored in Left$[j] =$ Up$[i]$. We map v to the point (i, j) and update the two arrays as explained below. In Case 2, we put a crossing in the point (i, j) and leave Left and Up unchanged. In Case 3, let $v =$ Left$[j]$; we map v to (i, j) and update the two arrays as described below. Similarly, in Case 4, let $v =$ Up$[i]$; we map v to (i, j) and update the two

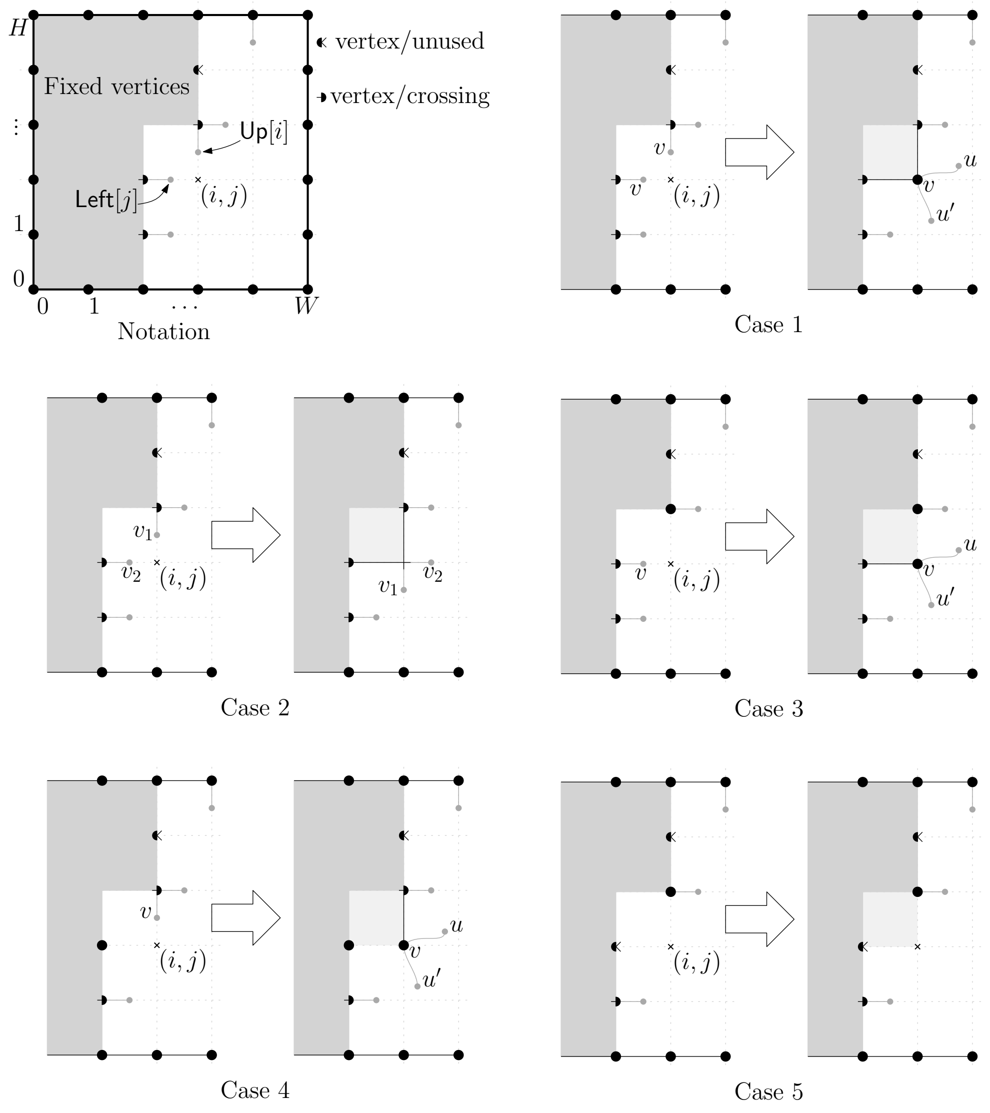


Figure 2 Theorem 3.1: Algorithm notation (top left) and Cases 1–5 when processing point (i, j).

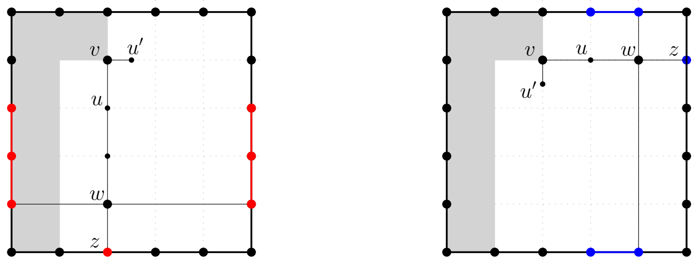


Figure 3 If z coincides with a red (blue) point of Γ_C, then $x(u) = x(v)$ $(y(u) = y(v))$.

arrays as described below. Finally, in Case 5 the point (i, j) will be left unused (it will be internal to a face), and we leave Left and Up unchanged.

The updates of the values Left$[j]$ and Up$[i]$ in Cases 1, 3, and 4, are done as follows. Let U be the set of neighbors of v that are not fixed. If $|U| = 0$, we reject the instance, as either $\deg_G(v) = 2$ and its incident edges form a 270° angle (Case 1), or $\deg_G(v) = 1$ (Cases 3 and 4). Also, if $|U| \geq 3$, we reject the instance as in this case there is no possibility of placing all vertices in U. Thus it must be $|U| \in \{1, 2\}$; let u and u' be the two vertices of U, possibly with $u' =$ null if $|U| = 1$. In this case, for each vertex u and u', either its coordinates are unassigned or it belongs to C. Suppose first that at least one of them, say u, belongs to C; if $x(v) \neq x(u)$ and $y(v) \neq y(u)$, we reject the instance because v and u cannot be horizontally or vertically aligned. If $x(v) = x(u)$ we set Up$[i] = u$ and Left$[j] = u'$; if $y(v) = y(u)$ we set Up$[i] = u'$ and Left$[j] = u$. If u' exists and it is already placed, then we further check that its coordinates are consistent with its assignment to Up$[i]$ or Left$[j]$.

Suppose now that neither u nor u' is in C. If $\deg_G(u) \geq 3$ we set $w{=}u$; Otherwise we follow the edge incident to u distinct from (u, v) and continue traversing all degree-2 vertices until we reach a degree-3 vertex w. If w is fixed, we reject the instance, as either the traversed path can only be drawn with an angle of 270°, or u has to be on the left of v, contradicting the fact that it is not fixed. Hence w is non-fixed and either it belongs to C or its coordinates are unassigned. Assume first that $w \in C$. If $x(w){\neq}x(v)$ and $y(w){\neq}y(v)$ we reject the instance, as the path visited by the traversal must be either horizontal or vertical. Otherwise, if $x(w){=}x(v)$ we set Up$[i]{=}u$, and Left$[j]{=}u'$. Similarly, if $y(w){=}y(v)$ we set Left$[j]{=}u$ and Up$[i]{=}u'$. If the coordinates of w have not been assigned, then $w \in G \setminus C$. Since $G \setminus C$ is a 2-graph and $\deg_G(w) > 2$, at least one neighbor of w, call it z, belongs to C. If z lies on the left or right side of Γ_C, and $0 < y(z) < y(v)$ or $x(z) = x(v)$ and $y(z) = 0$, we set Up$[i] = u$ and Left$[j] = u'$ (see Figure 3 (left)). If z lies on the top or bottom side of Γ_C, and $x(v) < x(z) < W$ or $x(z) = W$ and $y(z) = y(v)$, we set Left$[j] = u$ and Up$[i] = u'$ (see Figure 3(right)). If z does not lie in any of the described positions, we reject the instance, as z cannot share a coordinate with w, and thus the path from v to w cannot be drawn horizontally or vertically.

Once all the grid points have been considered, if a vertex of G has unassigned coordinates, we reject the instance, otherwise we have a drawing with external boundary Γ_C.

◀

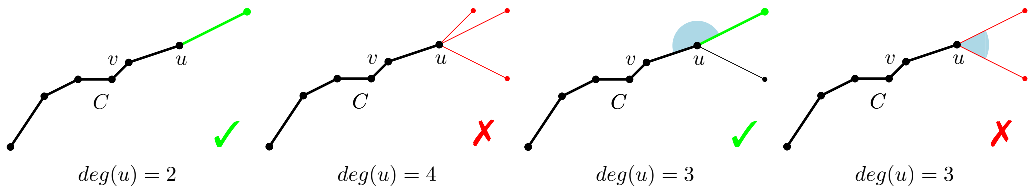


Figure 4 Candidate cycle construction: the green edge is the next edge of C.

4 General Case

For the general case we describe an FPT algorithm in the number of degree-3 vertices. A *large-angle assignment* of a 4-graph G defines, for each degree-3 vertex v of G, a *large angle*, i.e., the two edges incident to v that will form a 180° angle in any UER-RF drawing of G.

▶ **Theorem 4.1.** *Let G be a 4-graph and let k be the number of degree-3 vertices G. There exists an FPT algorithm, with parameter k, that tests whether G admits a UER-RF drawing. If such a drawing exists, the algorithm constructs one. Moreover, the algorithm can be adapted to preserve a given rotation system and/or to have a prescribed external cycle. The time complexity of the algorithm is: **1.** $O(3^k n^2)$ if the four corners are given; **2.** $O(3^k n^4)$ if the external cycle is prescribed; **3.** $O(3^k n^{4.5})$ in the general case.*

Proof sketch. We describe a primary algorithm (PA) that receives in the input graph G and a large-angle assignment. PA can be used to obtain an FPT algorithm for the general case by guessing for each degree-3 vertex a pair of incident edges to form the large angle. PA has two steps: the first selects and draws the external cycle, and the second draws the internal vertices.

Step 1. Given a 4-graph with a large-angle assignment, PA selects a set of cycles of G candidates to be the external boundary of a UER-RF drawing of G and, for each of them, a set of rectangles representing it: one for each possible choice of corners.

If the four corners are given (Case 1) the candidate cycle C is unique and is computed one edge at a time, starting from one corner. Let u be the last vertex added to C (thus, one of its edges, call it (v, u), has already been selected to be in C). See Figure 4. If $\deg_G(u) = 4$, we reject the instance and if $\deg_G(u) = 2$ we select the second edge incident to u. If $deg_G(u) = 3$, we select the edge that forms the large angle with (u, v) or we reject the instance if no such edge exists. In Cases 2 and 3 we suitably guess the candidate cycles and we prove that they are $O(n^2)$ in Case 2 and $O(n^{2.5})$ in Case 3.

Step 2. From Step 1 we obtain either one or $O(n^2)$ or $O(n^{2.5})$ rectangles, respectively, in the three cases of the statement. It is possible to place the vertices of $G \setminus C$ in the interior of one of these rectangles in $O(n^2)$ time. Similarly to Theorem 3.1, we process the internal grid points from left to right and from top to bottom. In this case, we choose whether each grid point is a vertex or a crossing, or is left empty, by exploiting the large-angle assignment of the vertices lying above and to the left of the considered point. ◀

5 Open Problems

The complexity of the problem in the general case remains open, as well as whether one can improve the complexity of our polynomial-time algorithms. Another interesting research direction is to extend our drawing convention to vertices of degree larger than four.

References

1 Zachary Abel, Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Jayson Lynch, and Tao B. Schardl. Who needs crossings? Hardness of plane graph rigidity. In *32nd Int. Symp. on Computational Geometry (SoCG '16)*, volume 51 of *LIPIcs*, pages 3:1–3:15, 2016. doi:10.4230/LIPIcs.SoCG.2016.3.

2 Carlos Alegría, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Fabrizio Grosso, and Maurizio Patrignani. Unit-length rectangular drawings of graphs. In *30th Int. Symp. on Graph Drawing and Network Visualization (GD'22)*, volume 13764 of *LNCS*, pages 127–143. Springer, 2022. doi:10.1007/978-3-031-22203-0_10.

3 Carlos Alegría, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Fabrizio Grosso, and Maurizio Patrignani. Unit-length rectangular drawings of graphs. *JGAA*, 28(1):403–437, 2024. doi:10.7155/jgaa.v28i1.2996.

4 Patrizio Angelini, Carla Binucci, Giuseppe Di Battista, Emilio Di Giacomo, Walter Didimo, Fabrizio Grosso, Giacomo Ortali, and Ioannis G. Tollis. Unit edge-length rectilinear drawings with crossings and rectangular faces. *CoRR*, 2503.01526, 2025. URL: http://arxiv.org/abs/2503.01526.

5 Sergio Cabello, Erik D. Demaine, and Günter Rote. Planar embeddings of graphs with specified edge lengths. In Giuseppe Liotta, editor, *11th Int. Symp. on Graph Drawing and Network Visualization (GD'03)*, volume 2912 of *Lecture Notes in Computer Science*, pages 283–294. Springer, 2003. doi:10.1007/978-3-540-24595-7_26.

6 Sergio Cabello, Erik D. Demaine, and Günter Rote. Planar embeddings of graphs with specified edge lengths. *JGAA*, 11(1):259–276, 2007. doi:10.7155/jgaa.00145.

7 Peter Eades and Nicholas C. Wormald. Fixed edge-length graph drawing is NP-hard. *Disc. Appl. Math.*, 28(2):111–134, 1990. doi:10.1016/0166-218X(90)90110-X.

8 Ashim Garg and Roberto Tamassia. On the compuational complexity of upward and rectilinear planarity testing. In Roberto Tamassia and Ioannis G. Tollis, editors, *DIMACS International Workshop (GD'94)*, volume 894 of *Lecture Notes in Computer Science*, pages 286–297. Springer, 1994. doi:10.1007/3-540-58950-3_384.

9 Ashim Garg and Roberto Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM J. Comput.*, 31(2):601–625, 2001. doi:10.1137/S0097539794277123.

10 Md. Saidur Rahman, Takao Nishizeki, and Shubhashis Ghosh. Rectangular drawings of planar graphs. In Stephen G. Kobourov and Michael T. Goodrich, editors, *10th Int. Symp. on Graph Drawing and Network Visualization (GD'02)*, volume 2528 of *Lecture Notes in Computer Science*, pages 244–255. Springer, 2002. doi:10.1007/3-540-36151-0_23.

11 Md. Saidur Rahman, Takao Nishizeki, and Shubhashis Ghosh. Rectangular drawings of planar graphs. *J. Algorithms*, 50(1):62–78, 2004. doi:10.1016/S0196-6774(03)00126-3.

12 Marcus Schaefer. Realizability of graphs and linkages. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 461–482. Springer New York, New York, NY, 2013. doi:10.1007/978-1-4614-0110-0_24.

13 Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Comput.*, 16(3):421–444, 1987. doi:10.1137/0216030.

Transforming Dogs on the Line: On the Fréchet Distance Under Translation in 1D

Lotte Blank[1], Jacobus Conradi[1], Anne Driemel[1,2], Benedikt Kolbe[1,2], André Nusser[3], and Marena Richter[1]

1 **University of Bonn, Germany**
`{lblank,bkolbe,marenarichter}@uni-bonn.de, {conradi,driemel}@cs.uni-bonn.de`

2 **Lamarr Institute for Machine Learning and Artificial Intelligence, Germany**

3 **Université Côte d'Azur, CNRS, Inria, France**
`andre.nusser@cnrs.fr`

Abstract

The Fréchet distance is a computational mainstay for comparing polygonal curves. The Fréchet distance under translation, which is a translation invariant version, considers the similarity of two curves independent of their location in space. It is defined as the minimum Fréchet distance that arises from allowing arbitrary translations of the input curves. This problem and numerous variants of the Fréchet distance under some transformations have been studied, with more work concentrating on the discrete Fréchet distance, leaving a significant gap between the discrete and continuous versions of the Fréchet distance under transformations. We present an algorithm for the Fréchet distance under translation on 1-dimensional curves of complexity n with a running time of $\mathcal{O}(n^{8/3}\log^3 n)$. We match the running times of the discrete case and improve the previously best known bounds of $\tilde{\mathcal{O}}(n^4)$. Our algorithm relies on technical insights but is conceptually simple, essentially reducing the continuous problem to the discrete case across different length scales.

1 Introduction

The Fréchet distance is one of the most well-studied distance measures for polygonal curves. An important aspect of detecting movement patterns in different application areas is to consider the movement independent of its absolute location and scale, i.e., we want to know for which location and scale do the patterns look most similar. Concretely, the Fréchet distance under translation is the minimum Fréchet distance that we obtain by allowing an arbitrary translation of one curve. The *continuous* Fréchet distance under translations was first studied in 2001 [1, 2, 6]. Shortly after, Wenk [7] developed the currently best published algorithm in d-dimensional Euclidean space. For 1-dimensional curves (i.e., time series), the algorithm achieves a running time of $\mathcal{O}(n^5 \log n)$. We note that for time series there is an unpublished $\tilde{\mathcal{O}}(n^4)$ algorithm that can be considered folklore.

In this paper, we show that the continuous Fréchet distance between two time series of complexity n can be computed in $\mathcal{O}(n^{8/3}\log^3 n)$ time. The result is made possible by the introduction of a novel framework for studying (continuous) time series [3]. Our approach essentially reduces (in an algorithmic sense) the continuous problem to its discrete counterpart and hence surprisingly matches the currently best result from [4] in the discrete setting.[1]

Missing proofs can be found in the full version.

[1] This is a result of the arrangement size in $\mathbb{R}$ being n^2, while in $\mathbb{R}^2$ it is n^4.

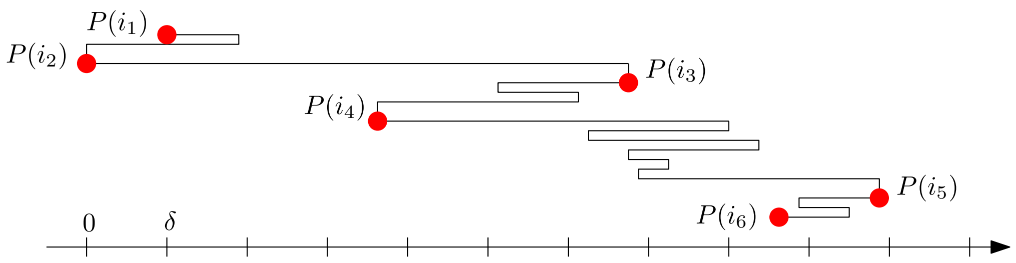


Figure 1 Throughout this paper, vertices of time series are drawn as vertical segments for clarity. The red vertices of the time series P are its δ-signature vertices.

2 Preliminaries

We denote with $[n]$ the set $\{1,\dots,n\}$. For any two points $p_1, p_2 \in \mathbb{R}$, $\overline{p_1p_2}$ is the directed line segment from p_1 to p_2. A *time series* of complexity n is a 1-dimensional curve formed by n *vertices* $P(1),\dots,P(n) \in \mathbb{R}$ and the ordered line segments, called *edges*, $\overline{P(i)P(i+1)}$ for $i = 1,\dots,n-1$. Such a time series can be viewed as a function $P : [1,n] \to \mathbb{R}$, where $P(i+\alpha) = (1-\alpha)P(i) + \alpha P(i+1)$ for $i = 1,\dots,n-1$ and $\alpha \in [0,1]$. We denote P also as $\langle P(1), P(2),\dots,P(n)\rangle$. For $1 \le s \le t \le n$, we denote by $P[s,t]$ the subcurve of P obtained by restricting the domain to the interval $[s,t]$. Further, we define $B(P,\delta) := \{x \mid \exists a \in [1,n] \text{ s.t. } |x - P(a)| \le \delta\}$ and $\operatorname{im}(P) := \{P(a) \mid a \in [1,n]\}$.

To define the Fréchet distance, let P and Q be two time series of complexity n and m. Further, let $\mathcal{H}_n$ be the set of all continuous non-decreasing functions $h : [0,1] \to [1,n]$ with $h(0) = 1$ and $h(1) = n$. The *continuous Fréchet distance* between P and Q is defined as

$$d_F(P,Q) = \min_{h_P \in \mathcal{H}_n, h_Q \in \mathcal{H}_m} \max_{\alpha \in [0,1]} |P(h_P(\alpha)) - Q(h_Q(\alpha))|.$$

For a time series $P = \langle P(1),\dots,P(n)\rangle$ and a value $t \in \mathbb{R}$, the *translated time series* P_t is $\langle P(1)+t,\dots,P(n)+t\rangle$ and the *Fréchet distance under translation* is defined as

$$d_F^T(P,Q) = \min_{t \in \mathbb{R}} d_F(P, Q_t).$$

To solve the decision problem for the Fréchet distance under translation, we use (slightly adapted) δ-signatures [3, 5], which are simplifications at the given length-scale δ and encode the large scale behavior of the time series in terms of a subset of selected vertices that essentially form a discrete time series. See Figure 1 for an example. Time series naturally decompose into three parts, the beginning, middle, and end, with the middle part given by the δ-signature. The task is to design and choose data structures for all parts in such a way that they can be efficiently updated and combined when the transformation changes.

▶ **Definition 2.1** (extended δ-signature). Let $P = \langle P(1),\dots,P(n)\rangle$ be a time series and $\delta \ge 0$. Then, an *extended δ-signature* $P' = \langle P(i_1),\dots,P(i_t)\rangle$ with $1 = i_1 \le i_2 < \dots < i_{t-1} \le i_t = n$ of P is a time series with the following properties:

(a) *(non-degenerate)* For $k = 2,\dots,t-1$, it holds that $P(i_k) \notin \overline{P(i_{k-1})P(i_{k+1})}$.
(b) *(2δ-monotone)* For $k = 1,\dots,t-1$, it holds that $P(s) \le P(s') + 2\delta$ for all $i_k \le s < s' \le i_{k+1}$ or $P(s) \ge P(s') - 2\delta$ for all $i_k \le s < s' \le i_{k+1}$.
(c) *(minimum edge length)* If $t > 4$, then for $k = 2,\dots,t-2$, $|P(i_k) - P(i_{k+1})| > 2\delta$.
(d) *(range)*
- For $k = 2,\dots,t-2$, it holds that $\operatorname{im}(P[i_k,i_{k+1}]) = \overline{P(i_k)P(i_{k+1})}$, and
- $\operatorname{im}(P[1,i_2]) \subset [P(i_2), P(i_2)+2\delta]$ or $\operatorname{im}(P[1,i_2]) \subset [P(i_2)-2\delta, P(i_2)]$, and
- $\operatorname{im}(P[i_{t-1},n]) \subset [P(i_{t-1}), P(i_{t-1})+2\delta]$ or $\operatorname{im}(P[i_{t-1},n]) \subset [P(i_{t-1})-2\delta, P(i_{t-1})]$.

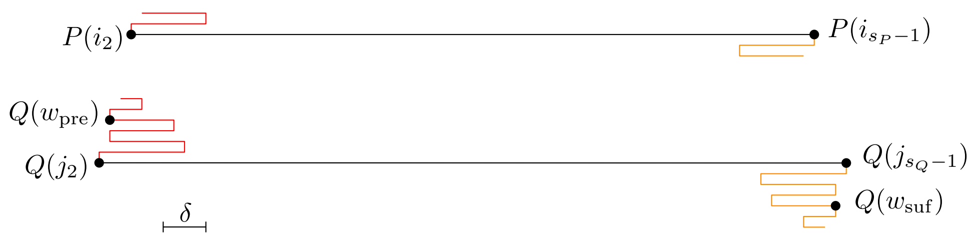


Figure 2 Minimal pre(P)-matcher on pre(Q), w_{pre}, and minimal suf(P)-matcher on suf(Q), w_{suf}.

The vertices of P' are called *δ-signature vertices* of P.

An extended δ-signature always exists and can be computed in $\mathcal{O}(n)$ time by [5].[2]

Grid Reachability. Let G be the directed $n \times n$-grid graph in which every node is either activated or deactivated. We are given updates $u_1, \dots, u_U$, which are of the form "activate node (i, j)" or "deactivate node (i, j)" in an offline manner. The task of *offline dynamic grid reachibility* is to compute for all $1 \le \ell \le U$ if (n, n) can be reached by $(1, 1)$ after updates $u_1, \dots, u_\ell$ are performed. Our main result follows by adapting everything so that we can use the theorem.

▶ **Theorem 2.2** (Theorem 3.1 of [4]). *Offline Dynamic Grid Reachability can be solved in time $\mathcal{O}(n^2 + Un^{2/3} \log^2 n)$.*

3 The Static Algorithm

In this section, we define the *modified free-space matrix* (see Figure 3 for an example) and discuss how it can be used to solve the decision problem of the Fréchet distance. Lemma 9 of [3] implies that, except for the beginning and the end, it is enough to look at pairs $(P(i), Q(j))$ of vertices, where one of the two vertices is a δ-signature vertex.

▶ **Definition 3.1.** Let P be a time series of complexity n and let $\langle P(i_1), P(i_2), \dots, P(i_{s_P})\rangle$ be its extended δ-signature. Then, the *prefix* of P is $\mathrm{pre}(P) := \langle P(1), P(2), \dots, P(i_2)\rangle$ and the *suffix* of P is $\mathrm{suf}(P) := \langle P(n), P(n-1), \dots, P(i_{s_P-1})\rangle$.

By the definition of extended δ-signatures, pre(P) and suf(P) are each contained in some δ-ball, i.e., there exist x and y such that $\mathrm{pre}(P) \subset B(x, \delta)$ and $\mathrm{suf}(P) \subset B(y, \delta)$.

▶ **Definition 3.2.** Let P and Q be time series of complexity n and m. The *minimal P-matcher on Q* is the smallest $w \in [m]$ such that $|P(n) - Q(w)| \le \delta$ and there is a $w^* \in [w, \min(w + 1, m)]$ with $\mathrm{d}_F(P, Q[1, w^*]) \le \delta$. If no such value exists, we set it to ∞.

We denote by w_{pre} the minimal pre(P)-matcher on pre(Q) and by v_{pre} the minimal pre(Q)-matcher on pre(P). For the suffix, we denote by w_{suf} the index of the vertex in Q corresponding to the minimal suf(P)-matcher on suf(Q) (and similarly v_{suf} when the roles of P and Q are swapped). See Figure 2 for an example. Due to the next observation, we will only discuss how to process the prefix in Section 3.1 and in the proofs of the lemmas of Section 4.

[2] This statement was proven for δ-signatures, but it can easily be shown for extended δ-signatures as well.

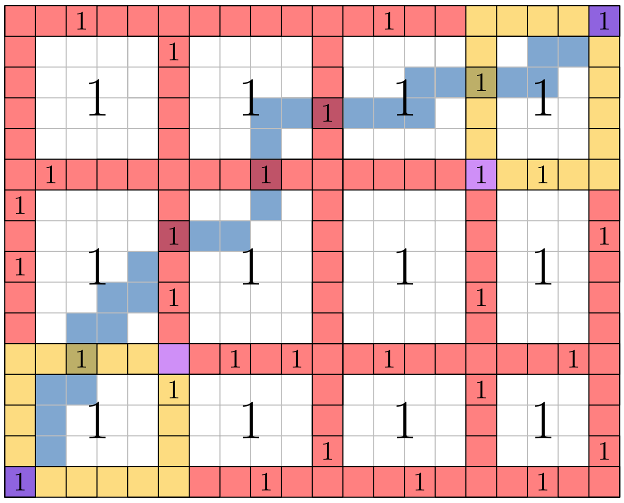


Figure 3 Example of a Modified Free-Space Matrix M_δ. The colored columns (resp. rows) correspond to the δ-signature vertices of P (resp. Q). The white entries are all 1 by Definition 3.4 a), the red entries are defined by b), the purple entries by c) and d) and e), and the yellow entries by f). The traversal drawn in blue uses only 1-entries of M_δ. Hence, $M_\delta(n,m)$ is reachable.

▶ **Observation 3.3.** *Let the minimal* $\text{pre}(\langle P(n),\dots,P(1)\rangle)$*-matcher on* $\text{pre}(\langle Q(m),\dots,Q(1)\rangle)$ *be* $\widehat{w_{\text{pre}}}$. *If m is the complexity of Q, then* $w_{\text{suf}} = m - (\widehat{w_{\text{pre}}} - 1)$.

▶ **Definition 3.4** (Modified Free-Space Matrix). Let P and Q be two time series of complexity n and m. Further, let $\langle P(i_1),\dots,P(i_{s_P})\rangle$ and $\langle Q(j_1),\dots,Q(j_{s_Q})\rangle$ be their extended δ-signatures. We construct a matrix $M_\delta \in \{0,1\}^{n\times m}$, where the entry $M_\delta(i,j) = 1$ if

a) $P(i)$ and $Q(j)$ are both not δ-signature vertices, or
b) $|P(i) - Q(j)| \le \delta$ and $(i,j) \notin [1,i_2]\times[1,j_2] \cup [i_{s_P-1},n]\times[j_{s_Q-1},m]$, or
c) $|P(i) - Q(j)| \le \delta$ and $(i,j) \in \{(1,1),(n,m)\}$, or
d) $|P(i) - Q(j)| \le \delta$ and $(i,j) = (i_2,j_2)$ and $d_F(\text{pre}(P),\text{pre}(Q)) \le \delta$, or
e) $|P(i) - Q(j)| \le \delta$ and $(i,j) = (i_{s_P-1},j_{s_Q-1})$ and $d_F(\text{suf}(P),\text{suf}(Q)) \le \delta$, or
f) $(i,j) \in \{(i_2,w_{\text{pre}}),(v_{\text{pre}},j_2),(i_{s_P-1},w_{\text{suf}}),(v_{\text{suf}},j_{s_Q-1})\}$.

Otherwise, the entry $M_\delta(i,j) = 0$. Further, we say that an entry $M_\delta(i,j)$ is reachable if there exists a traversal $(1,1) = (a_1,b_1),(a_2,b_2),\dots,(a_k,b_k) = (i,j)$ such that $M_\delta(a_l,b_l) = 1$ and $(a_l,b_l) \in \{(a_{l-1}+1,b_{l-1}),(a_{l-1},b_{l-1}+1),(a_{l-1}+1,b_{l-1}+1)\}$ for all $l = 2,\dots,k$.

The next lemma shows the importance of this matrix and follows mainly by a result in [3].

▶ **Lemma 3.5.** *It holds that $d_F(P,Q) \le \delta$ if and only if $M_\delta(n,m)$ is reachable. This can be checked in $\mathcal{O}(nm)$ time.*

3.1 Prefix and Suffix

To identify the minimal $\text{pre}(P)$-matcher on $\text{pre}(Q)$, we introduce another simplification. The crux is that we need to be able to do this efficiently for different transformations. We achieve this by introducing the structural notion of deadlocks. We often use the fact that $\text{pre}(P)$ and $\text{pre}(Q)$ are each contained in some δ-ball.

▶ **Definition 3.6.** Let P be a time series of complexity n, where $P(n)$ is a global extremum. The *extreme point sequence* of P is a sequence of indices $1 = a_1 < \dots < a_p = n$, such that

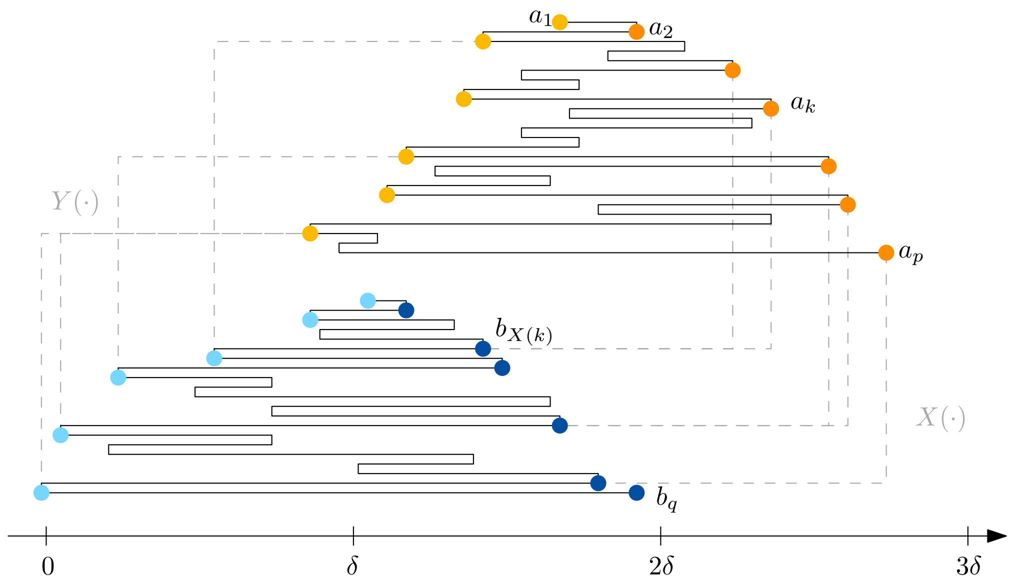


Figure 4 An extreme point sequence of the depicted time series P is $1 = a_1 < a_2 < \cdots < a_p$. The gray dashed lines mark the preliminary assignments $X(\cdot)$ and $Y(\cdot)$.

- (range-preserving) $\operatorname{im}(P[1, a_{k+1}]) = \overline{P(a_k)P(a_{k+1})}$ for all $k \in [p-1]$, and
- (extreme point) $P(a_k) = \min(P[1, a_k])$ or $P(a_k) = \max(P[1, a_k])$.

The next definition is the central notion to find the minimal pre(P)-matcher on pre(Q).

▶ **Definition 3.7.** Let P and Q be time series and $a_1 < \ldots < a_p$ and $b_1 < \ldots < b_q$ be extreme point sequences of P and Q respectively. Define the *preliminary assignment* $X(k)$ of P on Q for every $k \leq p$ to be the smallest index such that $|P(a_k) - Q(b_{X(k)})| \leq \delta$. If no such index exists, we set $X(k) = \infty$. Let similarly $Y(l)$ be the preliminary assignment of Q on P. We say that $X(k)$ and $Y(l)$ form a *deadlock*, if $l < X(k)$ and $k < Y(l)$.

See Figure 4 for an example of an extreme point sequence and their associated preliminary assignments. The importance of deadlocks is summarized in the following pivotal lemmas.

▶ **Lemma 3.8.** *Let P and Q be time series each contained in some δ-ball and $a_1 < \ldots < a_p$ and $b_1 < \ldots < b_q$ be extreme point sequences of P and Q. For any $w^* \leq b_q$, it holds that $d_F(P[1, a_p], Q[1, w^*]) \leq \delta$ if and only if*

(i) $|P(1) - Q(1)| \leq \delta$ *and* $|P(a_p) - Q(w^*)| \leq \delta$,
(ii) $\operatorname{im}(P[1, a_p]) \subset B(Q[1, w^*], \delta)$,
(iii) $\operatorname{im}(Q[1, w^*]) \subset B(P[1, a_p], \delta)$, *and*
(iv) *$X(k)$ and $Y(l)$ do not form a deadlock for all $k \in [p]$ and $l \in [q]$.*

▶ **Lemma 3.9.** *Let P and Q be time series and $P(i_2)$ be the second δ-signature vertex of P and $Q(j_2)$ of Q. There exists a minimal* pre(P)*-matcher on* pre(Q) *if and only if $X(k) < \infty$, and $X(k)$ and $Y(l)$ do not form a deadlock for all $k \in [p]$ and $l \in [q]$. If it exists, it is*

$$\min\{w = 1, \ldots, j_2 \mid |P(i_2) - Q(w)| \leq \delta,\ \operatorname{im}(\operatorname{pre}(P)) \subset B(Q([1, w+1], \delta)\}.$$

▶ **Lemma 3.10.** *Let P and Q be time series and let $\operatorname{im}(Q[1, j])$ be given for every $j \in [q]$. If the preliminary assignments X and Y of* pre(P) *and* pre(Q) *do not form a deadlock, we can compute the minimal* pre(P)*-matcher on* pre(Q) *and decide whether $d_F(\operatorname{pre}(P), \operatorname{pre}(Q)) \leq \delta$ in $\mathcal{O}(\log n)$ time.*

4 1D Continuous Fréchet Distance Under Translation

We identify and precompute a set of $\mathcal{O}(n^2)$ representative transformations at which the answer to the decision problem is subject to change, and for which the answer can be updated efficiently as we sweep over them. For that, we use the modified free-space matrix and Lemma 3.10. As the best algorithm for the discrete Fréchet distance under translation, we then make use of Theorem 2.2.

▶ **Lemma 4.1.** *There exist a sorted set $\mathfrak{T} \subset \mathbb{R}$ containing $\mathcal{O}(n^2)$ points (called* translation representatives*) and computable in $\mathcal{O}(n^2 \log n)$ time with the following properties.*

- *It holds that $d_F^T(P,Q) \leq \delta$ if and only if $\exists t \in \mathfrak{T}$ such that $d_F(P,Q_t) \leq \delta$.*
- *For two consecutive t, t' in $\mathfrak{T}$, there exist only one pair of indices $k, l \in [n]$ such that $|P(k) - Q_t(l)| \leq \delta$ and $|P(k) - Q_{t'}(l)| > \delta$, or the other way round. Further, for the set of all pairs of consecutive $t, t' \in \mathfrak{T}$, those indices can be computed in $\mathcal{O}(n^2 \log n)$ time.*

▶ **Lemma 4.2.** *There is an algorithm that correctly computes the set $\mathcal{T}$ of translations $t \in \mathfrak{T}$ for which the preliminary assignment of minimal* $\text{pre}(P)$*-matcher on* $\text{pre}(Q_t)$ *and the preliminary assignment of minimal* $\text{pre}(Q_t)$*-matcher on P do not form a deadlock in $\mathcal{O}(n^2 \log n)$. Similarly, we can compute the set of translations in $\mathfrak{T}$ for which* $\text{suf}(P)$ *and* $\text{suf}(Q_t)$ *do not form a deadlock in $\mathcal{O}(n^2 \log n)$ time.*

We apply Theorem 2.2 to maintain reachability in the modified free-space matrix for all translations in $\mathfrak{T}$ from Lemma 4.1. For the prefix and suffix, we use Lemma 3.10 and Lemma 4.2. Finally, parametric search turns the decision algorithm to an optimization algorithm.

▶ **Theorem 4.3.** *There exists an algorithm to compute the continuous Fréchet distance under translation between two time series of complexity n in time $\mathcal{O}(n^{8/3} \log^3 n)$.*

References

1 Helmut Alt, Christian Knauer, and Carola Wenk. Matching polygonal curves with respect to the Fréchet distance. In Afonso Ferreira and Horst Reichel, editors, *STACS 2001, 18th Annual Symposium on Theoretical Aspects of Computer Science, Dresden, Germany, February 15-17, 2001, Proceedings*, volume 2010 of *Lecture Notes in Computer Science*, pages 63–74. Springer, 2001. doi:10.1007/3-540-44693-1_6.

2 Helmut Alt, Christian Knauer, and Carola Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38(1):45–58, 2004. doi:10.1007/S00453-003-1042-5.

3 Lotte Blank and Anne Driemel. A faster algorithm for the Fréchet distance in 1d for the imbalanced case. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, September 2-4, 2024, Royal Holloway, London, United Kingdom*, volume 308 of *LIPIcs*, pages 28:1–28:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ESA.2024.28.

4 Karl Bringmann, Marvin Künnemann, and André Nusser. Discrete Fréchet distance under translation: Conditional hardness and an improved algorithm. *ACM Trans. Algorithms*, 17(3):25:1–25:42, 2021. doi:10.1145/3460656.

5 Anne Driemel, Amer Krivošija, and Christian Sohler. Clustering time series under the Fréchet distance. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 766–785. SIAM, 2016. URL: https://doi.org/10.1137/1.9781611974331.ch55, doi:10.1137/1.9781611974331.CH55.

6 Alon Efrat, Piotr Indyk, and Suresh Venkatasubramanian. Pattern matching for sets of segments. In *Proc. 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*, pages 295–304, 2001.

7 Carola Wenk. *Shape matching in higher dimensions.* PhD thesis, Freie Universität Berlin, 2002. PhD Thesis.

Guarding a 1.5D terrain with Imprecise Viewpoints[*]

Vahideh Keikha[1], Maarten Löffler[2], Maria Saumell[3], and Pavel Valtr[4]

1 **The Czech Academy of Sciences, Institute of Computer Science, Czech Republic**
`keikha@cs.cas.cz`

2 **Department of Information and Computing Sciences, Utrecht University, Netherlands**
`M.Loffler@uu.nl`

3 **Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, Czech Republic**
`maria.saumell@fit.cvut.cz`

4 **Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic**
`valtr@kam.mff.cuni.cz`

Abstract

Given an n-vertex 1.5D terrain $\mathcal{T}$ and a set of m edges of $\mathcal{T}$, we study the problem of placing one viewpoint on each edge so that the total length of the visible portions of the terrain is maximized. We present an $O(n + m \log m)$ time $\frac{1}{2}$-approximation algorithm for the general problem, and polynomial-time algorithms for the cases $m = 1$ and $m = 2$. Additionally, we show that the problem of computing a point on $\mathcal{T}$ maximizing the visible portion of $\mathcal{T}$ can be solved in $O(n^3)$ time.

1 Introduction

Visibility problems on terrains have been thoroughly studied in the literature. In this paper, we focus on *1.5D terrains*, defined as x-monotone polygonal lines in $\mathbb{R}^2$. Despite the extensive literature on this topic, a natural variant has not been considered: What happens if guards (also called *viewpoints*) are *imprecise*, i.e., their exact location is not known, but it is contained in prescribed regions of the terrain? Formally, let R be a set of m regions, each associated with one imprecise viewpoint. A point set $\mathcal{P}$ is a *realization* of R if there exists a bijection between $\mathcal{P}$ and R such that each point in $\mathcal{P}$ is contained in the corresponding region in R (the point is called the *representative* of the region). Since many different realizations are possible, natural optimization problems arise. The general problem we consider is:

***IV-mL* (m Imprecise Viewpoints on set L):** Given a 1.5D terrain $\mathcal{T}$ and a set L of m edges of $\mathcal{T}$, find a realization $\mathcal{P}$ of L such that the total length of the portion of $\mathcal{T}$ visible by $\mathcal{P}$ is maximized.

In this paper, we present an $O(n + m \log m)$ time $\frac{1}{2}$-approximation algorithm for *IV-mL*. Furthermore, we consider the special cases $m = 1$ and $m = 2$. Finally, we show that with our techniques we can solve a very natural and fundamental problem that does not seem to have been solved before: Computing a point on $\mathcal{T}$ maximizing the visible portion of $\mathcal{T}$.

[*] V.K. was supported by the CAS PPPLZ grant L100302301, and the institutional support RVO: 67985807. M.S. and P.V. were supported by the Czech Science Foundation, grant number 23-04949X.

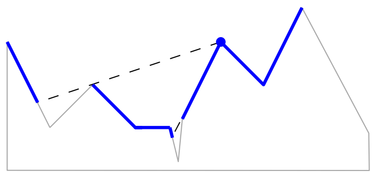


Figure 1 The viewshed of a viewpoint.

Related work. A common goal in this type of problems is to make the entire terrain visible. However, computing the minimum number of viewpoints on the terrain to achieve this is NP-hard [7]. If we have only one available viewpoint but it can be placed above the terrain (such a viewpoint is called a *watchtower*), computing a watchtower of minimum height that covers the entire terrain is an easy problem. The problem becomes more interesting when we can place two watchtowers. There are several versions of the problem depending on whether the basis of the watchtowers are required to lie on vertices of the terrain or not; all of the versions can be solved in polynomial time [1].

Another problem closely related to ours is that of constructing a data structure for reporting the visible portion of a terrain (or, more generally, a simple polygon) from a moving point along a given trajectory [4]. Additionally, an optimization problem generalizing ours (for fixed h, compute the watchtower at height h maximizing the visible portion of $\mathcal{T}$) has been considered in [5]. However, the algorithm is not correct (see Section 3.3).

The model of imprecision we adopt in this paper is called *region-based* and is the most common in the computational geometry literature. Regarding previous research on terrains in the region-based model, there are some studies where the elevation of the vertices of $\mathcal{T}$ is imprecise. Then the aim might be to optimize some terrain parameter [8], or to compute a shortest watchtower guarding the entire terrain [9].

Preliminaries. We assume that $\mathcal{T}$ does not have three collinear vertices. Let $V = \{v_1, \ldots, v_n\}$ denote its vertex set. The set of edges is $E = \{e_1, \ldots, e_{n-1}\}$, with $e_i = v_i v_{i+1}$. For each edge $e_i \in E$, let e_i^l and e_i^r denote the left and right endpoint of e_i, respectively.

For every vertex v_i, we say that v_i is *reflex* if the angle external to $\mathcal{T}$ between the edges incident to v_i is greater than π. If the angle is smaller than π, then the vertex is *convex*.

Two points on the terrain are said to be *visible* from each other if the line segment connecting them does not contain any point strictly below $\mathcal{T}$. The *viewshed* of a viewpoint p, denoted $\mathcal{V}_{\mathcal{T}}(p)$, is the union of all portions of $\mathcal{T}$ that are visible from p. See Fig. 1.

Omitted details and proofs will be given in the full version of the paper.

2 $\frac{1}{2}$-Approximation algorithm

Let $\mathcal{P}$ be a set of viewpoints on $\mathcal{T}$. We denote the union of the viewsheds of all viewpoints in $\mathcal{P}$ by $\text{Vis}^{\text{v}}(\mathcal{T}, \mathcal{P})$, and its length by $|\text{Vis}^{\text{v}}(\mathcal{T}, \mathcal{P})|$.

▶ **Lemma 2.1.** *Let p_i be a point of e_i. Then $\mathcal{V}_{\mathcal{T}}(p_i) \subseteq \mathcal{V}_{\mathcal{T}}(e_i^l) \cup \mathcal{V}_{\mathcal{T}}(e_i^r)$.*

▶ **Lemma 2.2.** *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two realizations of L such that, for each l_i, in $\mathcal{P}_1$ the representative of l_i lies on one of the endpoints of that edge and in $\mathcal{P}_2$ the representative of l_i lies on the other endpoint. Let $P^* = \{p_1^*, p_2^*, \ldots, p_m^*\}$ denote an optimal realization. Then,* $\max\{|\text{Vis}^{\text{v}}(\mathcal{T}, \mathcal{P}_1)|, |\text{Vis}^{\text{v}}(\mathcal{T}, \mathcal{P}_2)|\} \geq \frac{1}{2}|\text{Vis}^{\text{v}}(\mathcal{T}, P^*)|$.

Proof. We can assume that $l_i = e_i$ for $i = 1, 2, \ldots, m$. Then, $\text{Vis}^{\text{v}}(\mathcal{T}, P^*) = \cup_{i=1}^{m} \mathcal{V}_{\mathcal{T}}(p_i^*)$. By

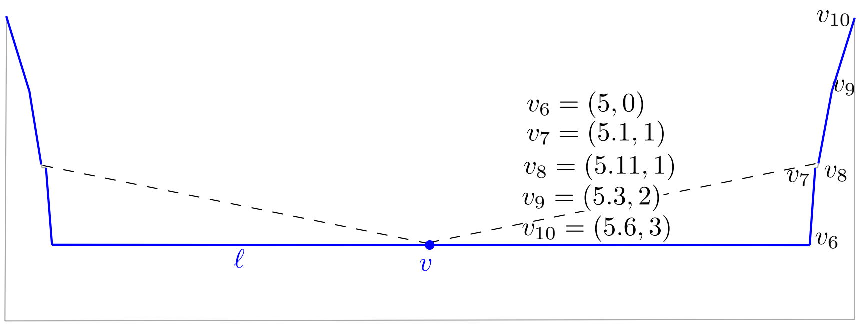


Figure 2 Example with one imprecise viewpoint on edge l. For the coordinates of vertices $v_1, v_2, \dots, v_5$, we note that the terrain is symmetric with respect to the y-axis. The point $v = (0, 0)$, which corresponds to the optimal placement inside l, lies in the interior of the edge.

Lemma 2.1, $\cup_{i=1}^m \mathcal{V}_\mathcal{T}(p_i^*) \subseteq \cup_{i=1}^m (\mathcal{V}_\mathcal{T}(e_i^l) \cup \mathcal{V}_\mathcal{T}(e_i^r)) = \mathrm{Vis}^\mathrm{v}(\mathcal{T}, \mathcal{P}_1) \cup \mathrm{Vis}^\mathrm{v}(\mathcal{T}, \mathcal{P}_2)$. Consequently,

$$|\mathrm{Vis}^\mathrm{v}(\mathcal{T}, P^*)| \le |\mathrm{Vis}^\mathrm{v}(\mathcal{T}, \mathcal{P}_1) \cup \mathrm{Vis}^\mathrm{v}(\mathcal{T}, \mathcal{P}_2)| \le 2\max\{|\mathrm{Vis}^\mathrm{v}(\mathcal{T}, \mathcal{P}_1)|, |\mathrm{Vis}^\mathrm{v}(\mathcal{T}, \mathcal{P}_2)|.$$

◀

The algorithm computes $|\mathrm{Vis}^\mathrm{v}(T, \mathcal{P}_1)|$ and $|\mathrm{Vis}^\mathrm{v}(\mathcal{T}, \mathcal{P}_2)|$ in $O(n + m \log m)$ time using the algorithm from [6], and takes the realization achieving the maximum among the two.

▶ **Theorem 2.3.** *For the IV-mL problem, a $\frac{1}{2}$-approximation can be computed in $O(n + m \log m)$ time.*

3 Case $m = 1$

We observe that the problem is not trivial because the optimal realization is not necessarily at an endpoint of the edge; see Fig. 2 for an illustrative example.

3.1 Preliminaries

Let e be the edge of $\mathcal{T}$ equal to the imprecise viewpoint. We parametrize e by $[0, 1]$. For each edge $e_x \in \mathcal{T}$ and each $\sigma \in [0, 1]$, we define $\mathcal{F}^x(\sigma) := |\mathcal{V}_\mathcal{T}(\sigma) \cap e_x|$.

Without loss of generality, we assume that e is to the left of e_x.

▶ **Lemma 3.1.** *The function $\mathcal{F}^x(\sigma)$ is non-increasing in $[0, 1)$.*

▶ **Definition 3.2.** Let e and e_x be edges of $\mathcal{T}$ such that e is to the left of e_x.

- Vertex τ *obstructs the visibility of e_x from point σ in e* if $\tau \notin \{e^r, e_x^l, e_x^r\}$ and the ray with origin at σ and passing through τ intersects e_x at a point q that is visible from σ.
- We denote by n_o^x the number of vertices τ of $\mathcal{T}$ such that there exists some σ in e satisfying that τ obstructs the visibility of e_x from point σ.

An example of a vertex obstructing the visibility of an edge is given in Fig. 3.

▶ **Lemma 3.3.** *Let $\sigma_0, \sigma_1 \in [0, 1]$, with $\sigma_0 < \sigma_1$, be two points in e such that, for any $\sigma \in [\sigma_0, \sigma_1]$, the visibility of e_x from σ is obstructed by the same vertex τ. Then,*

$$\mathcal{F}^x(\sigma) = \frac{A\sigma + B}{C\sigma + D}, \quad \textit{for } \sigma \in [\sigma_0, \sigma_1],$$

where A, B, C, D are functions of the coordinates of e, e_x and τ.

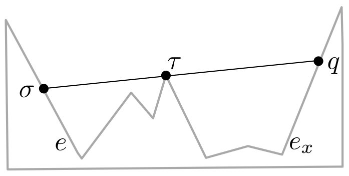


Figure 3 Vertex τ obstructs the visibility of e_x from point σ in e.

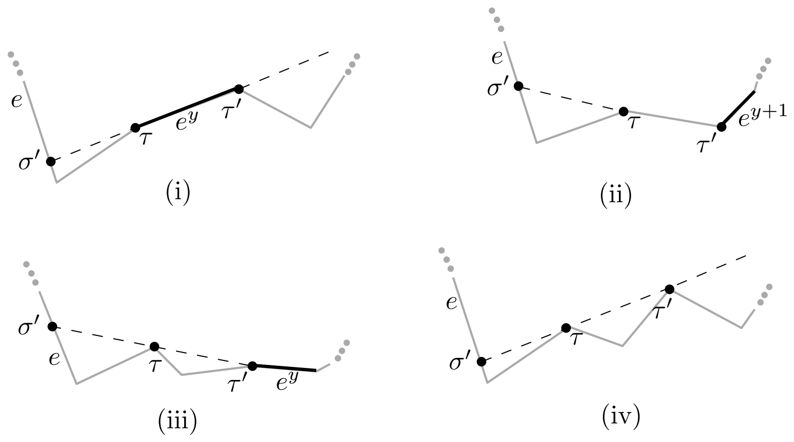


Figure 4 The four different types of critical points.

We are now ready to characterize the function $\mathcal{F}^x(\sigma)$.

▶ **Lemma 3.4.** *$\mathcal{F}^x(\sigma)$ is a piecewise-defined function. Each of the sub-functions has one of the following shapes:*
(i) ℓ*, with* $0 \leq \ell \leq |e_x|$*;*
(ii) $\frac{A\sigma+B}{C\sigma+D}$.
The number of pieces (i.e., sub-functions) of $\mathcal{F}^x(\sigma)$ is at most $n_o^x + 2$.

▶ **Definition 3.5.** A *critical point* of e is a point on e that is collinear with and is visible from two different vertices τ and τ' of $\mathcal{T}$ simultaneously such that τ and τ' are on the same side of the line that contains e.

Chen and Daescu [4] proved that the number of critical points of e is $O(n)$ (note that their result is for simple polygons). Together with further observations, this implies:

▶ **Theorem 3.6.** *The number of pieces of $\mathcal{F}^x(\sigma)$ added over all edges $e_x \in \mathcal{T}$ is $O(n)$.*

3.2 Computation of the functions $\mathcal{F}^x(\sigma)$

We sketch how to compute the functions $\mathcal{F}^x(\sigma)$ for all e_x to the right of e.

First, we compute $\mathcal{V}_{\mathcal{T}}(e^l)$ and $\mathcal{V}_{\mathcal{T}}(e^r)$. For every edge e_x such that e^r or e^l sees a portion of e_x that is not the whole e_x, we store the vertex obstructing the visibility of e_x (or e^r/e^l_x). Afterwards, we compute the set of critical points of e sorted along the edge. For every critical point we store the two vertices of $\mathcal{T}$ associated with it.

We classify the critical points into four types (see Fig. 4). We denote by σ' the critical point, by τ the leftmost terrain vertex associated with σ', and by τ' the rightmost such terrain vertex. The four types are based on the answers to: Is the portion of $\mathcal{T}$ immediately to the right of τ visible from σ'? And what about the portion immediately to the right of

τ'? If the answers are *yes, no*, the type is (i). If the answers are *yes, yes*, the type is (ii). If the answers are *no, yes*, the type is (iii). If the answers are *no, no*, the type is (iv).

In some cases, we can already determine some functions. For example, for type (i), τ and τ' are endpoints of the same edge e_y. Hence, we have that $\mathcal{F}^y(\sigma) = |e_y|$ for all $\sigma \in [0, \sigma']$ and $\mathcal{F}^y(\sigma) = 0$ for all $\sigma \in (\sigma', 1]$. In cases (i) and (iv), we compute via a ray-shooting query the first intersection after τ, τ' between $\mathcal{T}$ and the ray with origin at σ' and passing through τ, τ', and we assign the critical point σ' to the intersected edge (if any). In cases (ii) and (iii), we assign the critical point σ' to the edge with τ' as left endpoint.

After this preprocessing, we compute $\mathcal{F}^x(\sigma)$ for the remaining e_x. We start by sorting the critical points assigned to e_x. Note that $\mathcal{F}^x(0)$ and $\mathcal{F}^x(1)$ are obtained from $\mathcal{V}_\mathcal{T}(e^l)$ and $\mathcal{V}_\mathcal{T}(e^r)$, respectively. If $\mathcal{F}^x(0) = 0$, by Lemma 3.1 we are done. If e^r is reflex and e_x is the edge containing the first intersection of $\mathcal{T}$ with the ray with origin at e_l and passing through e_r, then $\mathcal{F}^x(\sigma)$ is constant for all $\sigma \in [0, 1)$.

Next, suppose that $\mathcal{F}^x(0) \in (0, |e_x|)$. Then there is a vertex τ obstructing the visibility of e_x from $\sigma = 0$ (τ has been found during the computation of $\mathcal{V}_\mathcal{T}(e^l)$). If e_x has critical points assigned to it, we take the first unprocessed one. For σ between 0 and the value of the critical point, the visibility of e_x is obstructed by τ. We compute $\mathcal{F}^x(\sigma)$ using Lemma 3.3. For values of σ infinitesimally greater than that of the critical point, the visibility of e_x is obstructed by the vertex associated to the critical point different from τ. Thus, we are in the same situation as in the beginning of this case. Otherwise, e_x has no unprocessed critical points assigned to it, so either the visibility of e_x is obstructed by τ until $\sigma = 1$, or the visibility of e_x is obstructed by τ until some value $\sigma' < 1$ and afterwards $\mathcal{F}^x(\cdot) = 0$. It is not difficult to distinguish between both cases and compute $\mathcal{F}^x(\sigma)$ in the rest of the domain.

The last case is when $\mathcal{F}^x(0) = |e_x|$. If $\mathcal{F}^x(1) = |e_x|$, it is easy to see that $\mathcal{F}^x(\sigma) = |e_x|$ for all $\sigma \in [0, 1]$. Otherwise, we deduce that there is a critical point σ' of type (ii) or (iii) whose two associated vertices are e^l_x and some vertex τ. The critical point corresponds to the first one assigned to e_x. For values of σ infinitesimally greater than σ', the visibility of e_x from σ is obstructed by τ. Thus, we are in the situation described in the previous paragraph.

We observe that the computation of the functions $\mathcal{F}^x(\sigma)$ for all e_x to the left of e is symmetric. We next analyze the total running time of the computation of all functions.

Computing $\mathcal{V}_\mathcal{T}(e^l)$ and $\mathcal{V}_\mathcal{T}(e^r)$ can be done in $O(n)$ time. Additionally, computing the set of critical points of e sorted along the edge can be done in $O(n \log n)$ [4].

During the preprocessing of critical points, for every critical point of types (i) or (iv), we perform a ray-shooting query in $O(\log n)$ time [3]. All other operations performed take $O(1)$ time. Since the number of critical points is $O(n)$, the total cost of this phase is $O(n \log n)$.

Afterwards, the functions $\mathcal{F}^x(\sigma)$ are calculated. Apart from initialization steps and possibly a ray-shooting query, the cost of computing of $\mathcal{F}^x(\sigma)$ takes constant time per critical point assigned to e_x. We conclude the following:

▶ **Theorem 3.7.** *The set of functions $\mathcal{F}^x(\sigma)$ can be computed in $O(n \log n)$ time.*

3.3 Optimization

By Theorem 3.6, $[0, 1]$ can be divided into a set $\mathcal{I}$ of $O(n)$ intervals such that, in every interval, each $\mathcal{F}^x(\sigma)$ is defined by only one piece. Let $[\sigma_0, \sigma_1] \in \mathcal{I}$. Since $\sum_{e_x \neq e} \mathcal{F}^x(\sigma)$ is potentially the sum of $\Theta(n)$ functions of type $\frac{A\sigma+B}{C\sigma+D}$, some of which are increasing and some of which are decreasing, it could in principle be possible that $\sum_{e_x \neq e} \mathcal{F}^x(\sigma)$ has $\Theta(n)$ local maxima in $\mathcal{I}$. By computing $(\mathcal{F}^x)''(\sigma)$, we can show that this is not the case.

▶ **Lemma 3.8.** $\sum_{e_x \neq e} \mathcal{F}^x(\sigma)$ *is either constant or strictly concave in each* $[\sigma_0, \sigma_1] \in \mathcal{I}$.

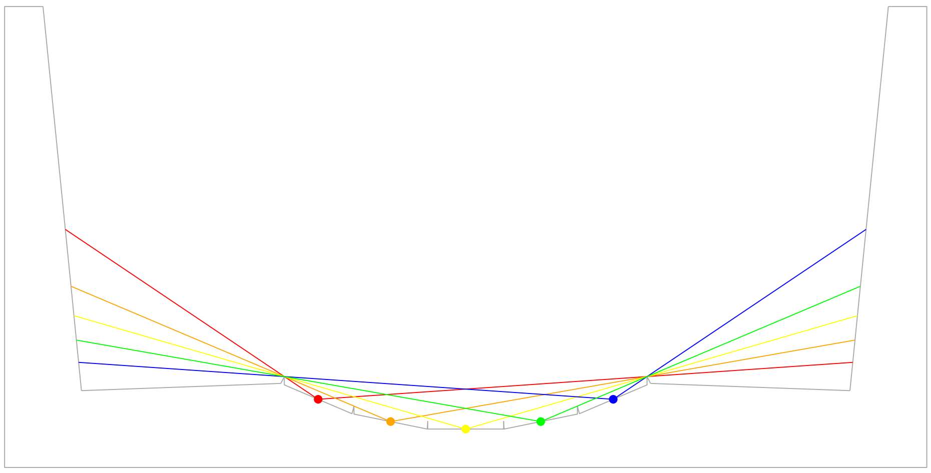


Figure 5 A terrain with $\Theta(n)$ edges e such that $\sum_{e_x \neq e} \mathcal{F}^x(\sigma)$ have a local maximum.

By Lemma 3.8, $\sum_{e_x \neq e} \mathcal{F}^x(\sigma)$ has at most one local maximum in $[\sigma_0, \sigma_1]$, so the global maximum is attained at this local maximum, at $\sigma = \sigma_0$ or at $\sigma = \sigma_1$. Since there does not seem to be an easy geometrical characterization of the local maximum (recall the example in Fig. 2), we solve $(\mathcal{F}^x)'(\sigma) = 0$. We assume that our model of computation is able to compute roots of polynomials of degree d in $O(d)$ time. Thus, we can solve $(\mathcal{F}^x)'(\sigma) = 0$ in $[\sigma_0, \sigma_1]$ in $O(n)$ time. Repeating this procedure for every interval in $\mathcal{I}$, we obtain the following result:

▶ **Theorem 3.9.** *The problem IV-1L can be solved in $O(n^2)$ time.*

If we run our algorithm for every edge e of the terrain, we can find a point on $\mathcal{T}$ that sees the largest portion of $\mathcal{T}$.

▶ **Theorem 3.10.** *We can compute a point on $\mathcal{T}$ maximizing the visible portion of $\mathcal{T}$ in $O(n^3)$ time.*

Computing such a point appears like a fundamental problem on visibility on terrains for which there only seems to be an incorrect algorithm: In [5], the authors study the following problem: Given a 1.5D terrain $\mathcal{T}$, find the location of a watchtower of given height h maximizing the visible portion of $\mathcal{T}$. Their approach is as follows: Let $\mathcal{T}'$ be the upward translation of $\mathcal{T}$ by height h. For each reflex vertex τ, a *grazing ray* is a ray originated at τ and extending upward following the direction of one of the two terrain edges incident to τ. The authors claim that the candidates points for the base of the optimal watchtower are the terrain vertices and the vertical projection on $\mathcal{T}$ of the intersections between $\mathcal{T}'$ and the grazing rays. However, if we consider the terrain in Fig. 2 and set up $h = 0.1$ (or smaller), the base of the optimal watchtower is at the origin, but there is no candidate at the origin as defined in the algorithm.

We finally show an example where there are many local maxima. The terrain, illustrated in Fig. 5, has $\Theta(n)$ edges e such that $\sum_{e_x \neq e} \mathcal{F}^x(\sigma)$ have a local maximum and each of these edges has $\Theta(n)$ pieces in the description of $\mathcal{F}^x$ added over all edges e_x.

4 *IV-2L*

We give the main ideas for this variant. Suppose that the imprecise viewpoints are located on edges e_i and e_j with $i < j$. We parameterize them by σ and δ, respectively. Thus, every realization can be seen as a pair $(\sigma, \delta) \in [0, 1] \times [0, 1]$.

For each edge $e_x \in \mathcal{T}$, we define $\mathcal{F}_i^x(\sigma) := |\mathcal{V}_{\mathcal{T}}(\sigma) \cap e_x|$ and $\mathcal{F}_j^x(\delta) := |\mathcal{V}_{\mathcal{T}}(\delta) \cap e_x|$. Recall that $[0,1]$ can be divided into a set $\mathcal{I}$ (resp. $\mathcal{J}$) of $O(n)$ intervals such that, in every interval, each $\mathcal{F}_i^x(\sigma)$ (resp. $\mathcal{F}_j^x(\delta)$) is defined by only one piece. Let $I \in \mathcal{I}$ and $J \in \mathcal{J}$. By doing a case analysis, we can see that the domain $I \times J$ might get divided by a curve into at most two portions such that the function giving $\mathcal{F}_{i,j}^x(\sigma, \delta) := |(\mathcal{V}_{\mathcal{T}}(\sigma) \cup \mathcal{V}_{\mathcal{T}}(\delta)) \cap e_x|$ within each portion is the same. We repeat this procedure for all edges e_x of $\mathcal{T}$ and consider the obtained arrangement of curves in $I \times J$, which has $O(n^2)$ faces. To obtain the value maximizing $\sum_{e_x} \mathcal{F}_{i,j}^x(\sigma, \delta)$ in $I \times J$, we traverse the faces of the arrangement one by one. We repeat this procedure for all pairs I, J with $I \in \mathcal{I}$ and $J \in \mathcal{J}$.

▶ **Theorem 4.1.** *The problem IV-2L can be solved in polynomial time.*

Acknowledgments. We would like to thank an anonymous reviewer for pointing out an issue in our previous conterexample.

References

1 Pankaj K Agarwal, Sergey Bereg, Ovidiu Daescu, Haim Kaplan, Simeon Ntafos, Micha Sharir, and Binhai Zhu. Guarding a terrain by two watchtowers. *Algorithmica*, 58(2):352–390, 2010. 10.1007/s00453-008-9270-3.

2 Boaz Ben-Moshe, Matthew J Katz, and Joseph SB Mitchell. A constant-factor approximation algorithm for optimal 1.5D terrain guarding. *SIAM Journal on Computing*, 36(6):1631–1647, 2007. 10.1137/S0097539704446384.

3 Bernard Chazelle, Herbert Edelsbrunner, Michelangelo Grigni, Leonidas Guibas, John Hershberger, Micha Sharir, and Jack Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994. 10.1007/BF01377183.

4 Danny Z. Chen and Ovidiu Daescu. Maintaining visibility of a polygon with a moving point of view. *Information Processing Letters*, 65(5):269–275, 1998. 10.1016/S0020-0190(97)00211-1.

5 Laxmi Gewali and Binay Dahal. Algorithms for tower placement on terrain. In *16th International Conference on Information Technology-New Generations (ITNG 2019)*, pages 551–556. Springer, 2019. 10.1007/978-3-030-14070-0_77.

6 Ferran Hurtado, Maarten Löffler, Inês Matos, Vera Sacristán, Maria Saumell, Rodrigo I. Silveira, and Frank Staals. Terrain visibility with multiple viewpoints. *International Journal of Computational Geometry & Applications*, 24(04):275–306, 2014. 10.1142/S0218195914600085.

7 James King and Erik Krohn. Terrain guarding is NP-hard. *SIAM Journal on Computing*, 40(5):1316–1339, 2011. 10.1137/100791506.

8 Anna Lubiw and Graeme Stroud. Computing realistic terrains from imprecise elevations. *Computing in Geometry and Topology*, 2(2):3–1, 2023. 10.57717/cgt.v2i2.29.

9 Bradley McCoy, Binhai Zhu, and Aakash Dutt. Guarding precise and imprecise polyhedral terrains with segments. In *Proceedings of the 17th International Conference on Combinatorial Optimization and Applications*, pages 323–336, 2023. 10.1007/978-3-031-49614-1_24.

On the Voronoi Diagram of Four Lines in Three Dimensions[*]

Evanthia Papadopoulou, Martin Suderland, and Zeyu Wang

Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland
{evanthia.papadopoulou, martin.suderland, zeyu.wang}@usi.ch

Abstract

We present combinatorial and topological properties of Euclidean Voronoi diagrams of lines in 3-dimensions. We first express the exact number of edges, faces, and cells, as a function of vertices, in the Voronoi diagram of n lines. We then focus on the Voronoi diagram of four lines and show that the number of vertices in this diagram is always even, between 0 and 8, under mild general position assumptions. Further, if the diagram has no vertex, then it has a unique topology; if it has two vertices, then it can have five topologies. If the diagram has more than two vertices then there is a *twist* in the trisector system.

1 Introduction

Voronoi diagrams are among the most fundamental space partitioning structures in Computational Geometry. Given a set S of n objects in some spaces, called sites, the *nearest* (respectively, *farthest*) Voronoi diagram of S decomposes the underlying space into regions, that have the same closest (resp., farthest) site. In this paper, we consider Voronoi diagrams of lines in $\mathbb{R}^3$ under the Euclidean metric, including their nearest and farthest variants.

Voronoi diagrams in the Euclidean plane have been intensively studied [2]. In their standard form they are typically of linear complexity and can be efficiently computed in $O(n \log n)$ time for a large class of sites and metrics. In three dimensions, however, Voronoi diagrams are far less well-understood. Early attention was given to the Euclidean Voronoi diagram of points in d-dimensional space. Its complexity is $O(n^{\lceil \frac{d}{2} \rceil})$, it can be computed in $O(n \log n + n^{\lceil \frac{d}{2} \rceil})$, and these bounds are tight [5, 6, 10]. These results also hold for certain polyhedral norms, such as the L_∞ or L_1 norm [4, 9]. For the Euclidean farthest Voronoi diagram, exact worst-case bounds in the range of $\Theta(n^{\lceil \frac{d}{2} \rceil})$, have also been reported [13].

For non-point objects in $\mathbb{R}^3$, near-tight combinatorial bounds are known for only restricted cases. These include $O(c^3 n^{2+\varepsilon})$ for lines with a constant number of c orientations [11] and $O(n^{2+\varepsilon})$ for parallel halflines [1]. A numerically robust algorithm for computing the Voronoi diagram of lines in $\mathbb{R}^3$, which uses the CGAL envelope package has been illustrated by Hemmer et al. [8]. The unbounded features of the order-k Voronoi diagrams of lines and line segments in $\mathbb{R}^d$ have been studied by Barequet et al. [3]. They are encoded in the Gaussian map, a map on the sphere of directions, which has complexity $O(\min\{k, n-k\} n^{d-1})$.

In summary, the Euclidean Voronoi diagram of generalized sites in 3D is a challenging problem, and even the seemingly simple case of lines as sites is no exception. For many years, geometers have tried to close the complexity gap between the $\Omega(n^2)$ and $O(n^{3+\varepsilon})$ known combinatorial bounds [12]. The algebraic description of the features of the Voronoi diagram, especially its edges, becomes quite complicated, which is why an entire paper was devoted to characterizing the Euclidean Voronoi diagram of only three lines [7]. It is thus natural

[*] This research was supported by the Swiss National Science Foundation (SNF), project 200021E_201356. Martin Suderland was also supported by SNF project P500PT_206736/1.

to further investigate the Voronoi diagram of four lines in $\mathbb{R}^3$, which may induce a Voronoi vertex.

In this abstract, we list structural properties of the Voronoi diagram of four lines in $\mathbb{R}^3$. We first express the exact number of edges, faces, and cells, in the nearest and farthest Voronoi diagram of n lines as a function of number of vertices, see Table 1. In the farthest Voronoi diagram of lines, each line always has exactly $n-1$ cells and this is independent of the number of vertices in the diagram. Under some mild general position assumptions, we show that the number of vertices in the Voronoi diagram of 4 lines is always even, between 0 and 8, and all such numbers can be realized. Further, if the diagram has no vertex, then it has a unique topology; if it has two vertices, then it can have five different topologies. We identify a substructure in the trisector system, called a *twist*, which is necessary to exist, if the diagram has more than 2 vertices. A twist corresponds to a pair of intersections between two related trisector branches and gives the ability to study the diagram according to its number of vertices.

2 Preliminaries

Let $L = \{\ell_1, \cdots, \ell_n\}$ be a set of n lines in $\mathbb{R}^3$. We assume that the lines satisfy the following general position assumption: (1) no sphere is touching 5 lines; (2) no circle is touching 4 lines; (3) lines are pairwise skew, and no three lines are parallel to the same plane.

We denote by $d(x, y)$ the Euclidean distance between two points $x, y \in \mathbb{R}^3$. The distance $d(x, \ell)$ from a point $x \in \mathbb{R}^3$ to a line $\ell \in L$ is $d(x, \ell) = \min\{d(x, y) \mid y \in \ell\}$. The i-sector of i lines in $\mathbb{R}^3$ is the locus of points at equal distance from the i lines. For $i = 2$, a bisector $B(\cdot, \cdot)$ is a hyperbolic paraboloid; for $i = 3$, a trisector $T(\cdot, \cdot, \cdot)$ is a quartic consisting of four unbounded branches [7, 11]. Two trisectors are *related* if they are defined by four lines.

▶ **Definition 2.1.** For a subset of sites $H \subset L$ of size $|H| = k$, $1 \leq k < n$, the *order-k Voronoi region* of H is the set of points in $\mathbb{R}^3$ whose distance to any site in H is smaller than to any site not in H. It is denoted as $\mathrm{reg}(H, L) = \{p \in \mathbb{R}^3 \mid \forall h \in H, \forall \ell \in L \setminus H : d(p, h) < d(p, \ell)\}$.

The order-k regions of L induce a subdivision in $\mathbb{R}^3$; and the induced cell complex is the ***order-k Voronoi diagram*** of L, denoted by $\mathrm{VD}_k(L)$. If the set of sites L is clear, we may use notation VD_k and $\mathrm{reg}(H)$. When $k = 1$, $\mathrm{VD}_k(L)$ is the *nearest Voronoi diagram* of L, denoted by $\mathrm{NVD}(L)$. When $k = n-1$, it is the *farthest Voronoi diagram* of L, denoted by $\mathrm{FVD}(L)$. The cell complex partitions the space into features of the diagram: vertices, edges, faces, and 3D cells; we refer to the latter simply as cells. We say two cell complexes M and M' have the same ***topology*** if there is an isomorphism that maps the features of M to features of the same dimension in M' and maintains all incidence relations.

The general position assumptions ensure the following properties: by assumption (1), vertices of the Voronoi diagram have degree 4 in the NVD and FVD; by (2), related trisectors intersect transversely (Lemma 2.2); by (3), the topology of trisectors is unique [7].

▶ **Lemma 2.2.** *If two related trisectors are intersecting tangentially, then there exists a circle which is touching 4 lines.*

Bisectors of lines are homeomorphic to the Euclidean plane. Let $\Pi : B(\ell_i, \ell_j) \to \mathbb{R}^2$ be such a homeomorphism and let T be a trisector of the form $T(\ell_i, \ell_j, \ell_k)$. We are often interested in the *projected bisector* $\Pi(B(\ell_i, \ell_j))$ and the *projected trisector* $\Pi(T)$. Each projected trisector consists of 4 unbounded branches [7]. It has 2 vertical and 2 horizontal asymptotes. There is a unique branch that admits only one asymptote, called the *middle*

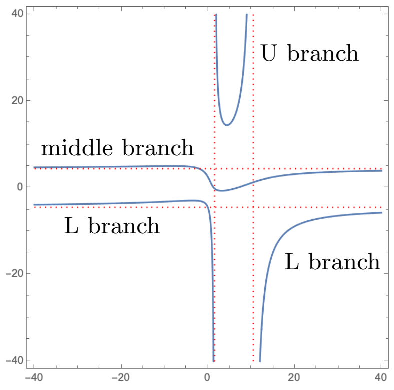


Figure 1 A projected trisector in blue. Its 4 asymptotes are in red; branch names are indicated.

branch. The middle branch partitions the bisector into two regions, one has a single branch, called the *U branch*; another has two branches called the *L branches*. Refer to Fig. 1.

▶ **Definition 2.3.** For a finite cell complex M, let B be a topological ball large enough to intersect any cell of M of any dimension in one connected component. Let Γ denote the boundary of B. The intersection $M \cap \Gamma$ is called the Γ***-map*** of M, denoted by $\Gamma\mathrm{M}(M)$.

3 Combinatorial properties of the Voronoi diagram of lines

We give some combinatorial properties of Voronoi diagrams of n lines in $\mathbb{R}^3$ as a function of their number of vertices V, summarized in Table 1.

▶ **Lemma 3.1.** *The number of edges, faces, and cells in* $\mathrm{NVD}(L)$ *and* $\mathrm{FVD}(L)$ *and their respective* Γ*-maps are as stated in Table 1. Results are also given for* $VD_2(L)$ *for* $n = 4$.

	vertices	edges	faces	cells	
$\Gamma\mathrm{M}(\mathrm{NVD}(L))$	$4n-4$	$6n-6$	$2n$	–	
$\mathrm{NVD}(L)$	V	$2V+2n-2$	$V+3n-3$	n	
$\Gamma\mathrm{M}(\mathrm{VD}_2(L_4))$	32	48	18	–	$\Big\}$ for $n=4$
$\mathrm{VD}_2(L_4)$	V	$4V+16$	$4V+24$	$V+9$	
$\Gamma\mathrm{M}(\mathrm{FVD}(L))$	$2n^2-2n-4$	$3n^2-3n-6$	n^2-n	–	
$\mathrm{FVD}(L)$	V	$2V+n^2-n-2$	$V+2n^2-2n-3$	n^2-n	

Table 1 Number of features in VDs and their Γ-maps of n lines as a function of their vertices V.

Next we consider the Voronoi diagram of four lines in $\mathbb{R}^3$. This diagram has at most 8 vertices [11] and their number is even, assuming lines are in general position.

▶ **Lemma 3.2.** *If related trisectors are intersecting transversely, then the number of vertices in the Voronoi diagram of four lines is an even number.*

In the case of four lines as sites, any intersection of related trisectors appears on all six bisectors, and also appears as a vertex of any order-k Voronoi diagrams.

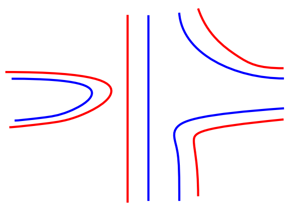


Figure 2 An impossible bisector configuration that violates Corollary 4.4.

4 The topology of the Voronoi diagram of four lines

In this section we study the Voronoi diagram of four lines $L_4 = \{\ell_1, \cdots, \ell_4\}$. There are six bisectors, each bisector contains exactly two trisectors; all trisectors are related. We call a *twist* the structure that occurs when two trisector branches intersect more than once.

▶ **Definition 4.1.** A pair of vertices v_1, v_2 is called a *twist on a bisector* B if the two related trisectors T_1, T_2 on B each have a branch C_1 of T_1 and C_2 of T_2 such that $\{v_1, v_2\} \subseteq C_1 \cap C_2$.

When the Voronoi diagrams have no twists, we are able to fully characterize their topologies and summarize in the following theorem:

▶ **Theorem 4.2.** *Consider the nearest (resp. farthest) Voronoi diagram of four lines in general position, and assume that there is no twist on any bisector. Then:*

(a) the diagram has a unique topology, if it contains no vertex;
(b) there are three different topologies, if the diagram has two vertices;
(c) this is not possible, if there are more than two vertices.

We outline the proof of the theorem in the remainder of this section. To this goal, we need some definitions. A projected trisector induces a map in the plane. We define a *configuration* to be the overlay of two such projected trisector maps, with one map's edges in red and the other's in blue. We say that two configurations are of the same type, if the underlying cell complexes have the same topology and the isomorphism also maintains colors and branch identities. We call a configuration C realizable, if there exist four lines whose configuration on a bisector is of the same type as C.

▶ **Lemma 4.3.** *The four trisectors associated with four lines contribute to the* Γ*-map of the* NVD *either* $0, 4, 4, 4$ *vertices respectively or* $2, 2, 2, 6$ *vertices respectively.*

We call a bisector an (i, j)*-bisector* if the two trisectors on it contribute i and j vertices each to the Γ-map of the NVD.

▶ **Corollary 4.4.** *With respect to the NVD, four lines induce either three* $(2, 2)$*-bisectors and three* $(2, 6)$*-bisectors, or three* $(0, 4)$*-bisectors and three* $(4, 4)$*-bisectors.*

The corollary shows that some bisector configurations are impossible, see e.g., Fig. 2.

To prove Theorem 4.2, we first assume that NVD(L_4) contains no vertex. Fig. 3 lists all bisector configuration types, without vertex, that satisfy Corollary 4.4, called type (a) and type (b). Then, by [7][Theorem 1], there can be three types of NVD faces (in green), type-A, B, and C. Each of the six bisectors is of configuration type (a) or (b). The next lemma shows that there is only one combination of bisector configurations that satisfies Lemma 4.3 and properties of trisectors pointed out in [7].

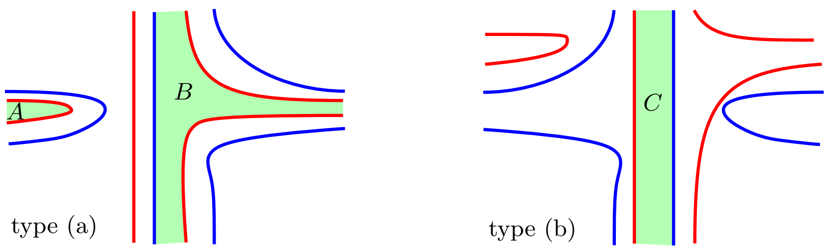


Figure 3 The two bisector configuration types with no vertex. Left: type (a); right: type (b). The NVD faces are shown in green; there are 3 types: A, B, C.

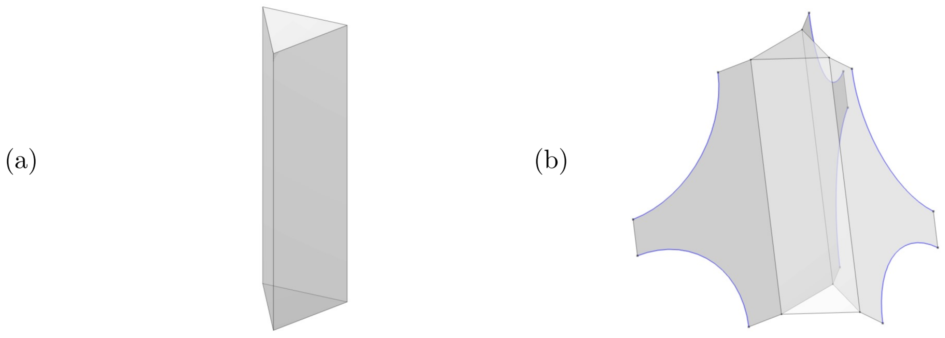


Figure 4 (a): after attaching type-C faces. (b): after further attaching type-B faces.

▶ **Lemma 4.5.** *If the Voronoi diagram of four lines has no vertex, then up to permutation of the lines, there is exactly one realizable combination of bisector configurations: bisectors $B(\ell_2, \ell_3), B(\ell_2, \ell_4), B(\ell_3, \ell_4)$ are of configuration type (a) and $B(\ell_1, \ell_2), B(\ell_1, \ell_3), B(\ell_1, \ell_4)$ are of configuration type (b).*

Let the bisector configurations types be as specified in Lemma 4.5. This fixes all edges and faces of the NVD. Every edge is incident to exactly three faces, each from a different bisector the edge is lying on.

We can now show that the topology of the Voronoi diagram is unique because there is a unique way of attaching the adjacent faces along their common edges to obtain the 3D diagram. In this process, we use properties of trisectors stated in [7], the fact that in NVD(L), each line has a connected, star-shaped cell, and that the NVD of three lines is unique, by [7][Theorem 1]. Next, we present the attaching steps.

Attaching type-C faces. There are three type-C face on $B(\ell_1, \ell_2), B(\ell_1, \ell_3), B(\ell_1, \ell_4)$ each. The outcome of attaching them is an infinite triangular prism as shown in Fig. 4(a). Note that $\mathrm{reg}(\ell_1, L_4)$ is bounded by these three faces and is the interior of the triangular prism.

Attaching type-B faces. Next, we attach the three type-B faces from $B(\ell_2, \ell_3)$, $B(\ell_2, \ell_4)$, $B(\ell_3, \ell_4)$. Using properties of the projected trisectors and the fact that the resulting diagram is a Voronoi diagram, we obtain the structure that is illustrated in Fig. 4(b).

Attaching type-A faces. There are three type-A faces from $B(\ell_2, \ell_3)$, $B(\ell_2, \ell_4)$, $B(\ell_3, \ell_4)$ that remain to be attached along three L/U branches of $T(\ell_2, \ell_3, \ell_4)$. Line ℓ_1 is not involved here. Hence, locally the topology is the same as NVD($\{\ell_2, \ell_3, \ell_4\}$), which is unique by [7].

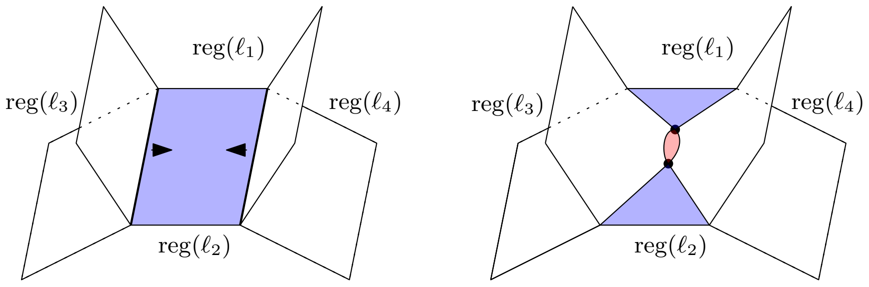


Figure 5 A full twist. Left: before; right: after.

In conclusion, there is at most one topology for the NVD of four lines without vertex and Fig. 6 shows an example that realizes it. The topology of the FVD can be obtained similarly.

We can prove Theorem 4.2(b) and (c) in a similar manner. As a corollary to Theorem 4.2(c), we have the following:

▶ **Corollary 4.6.** *If the Voronoi diagram of four lines has more than two vertices, then there must be at least one twist on some bisectors.*

4.1 Some observations and properties of twists

We show a special structure that can be viewed as the result of a dynamic process, refer to Fig. 5. Consider a face on $B(\ell_1, \ell_2)$ (in blue) and two edges incident to it (bold) that belong to two related trisectors $T(\ell_1, \ell_2, \ell_3), T(\ell_1, \ell_2, \ell_4)$ (see Fig. 5 left). Move the edges closer until they intersect twice (see Fig. 5 right). Locally, the blue face is split into two, and a new face (in red) is created, bounding reg(ℓ_3, L_4) and reg(ℓ_4, L_4) that previously were not touching. Such a process adds two vertices that form twists on all six bisectors. We call the resulting substructure a *full twist*. We call this process "adding a full twist on the blue face". Full twists commonly appear in diagrams that contain vertices. In fact, we can show the following theorem.

▶ **Theorem 4.7.** *If the* NVD *of four lines has two vertices and there exist twists, then there are two topologies. Each topology is obtained by adding a full twist to the* NVD *with no vertex on a face of type-B or on a face of type-C, which are shown in Fig. 3.*

▶ **Corollary 4.8.** *If the* NVD *of four lines has two vertices, then it can attain five topologies; three of them have no twists and two of them have full twists.*

Whenever a full twist appears in a diagram, one could locally remove it, and focus on the topology of the remaining diagram with two fewer vertices. However, twists do not always look like a full twist when there are more than two vertices. There are examples where two vertices form a twist only on one bisector.

Identifying the topology of the Voronoi diagram of lines with four or more vertices and twists remains a topic for future research.

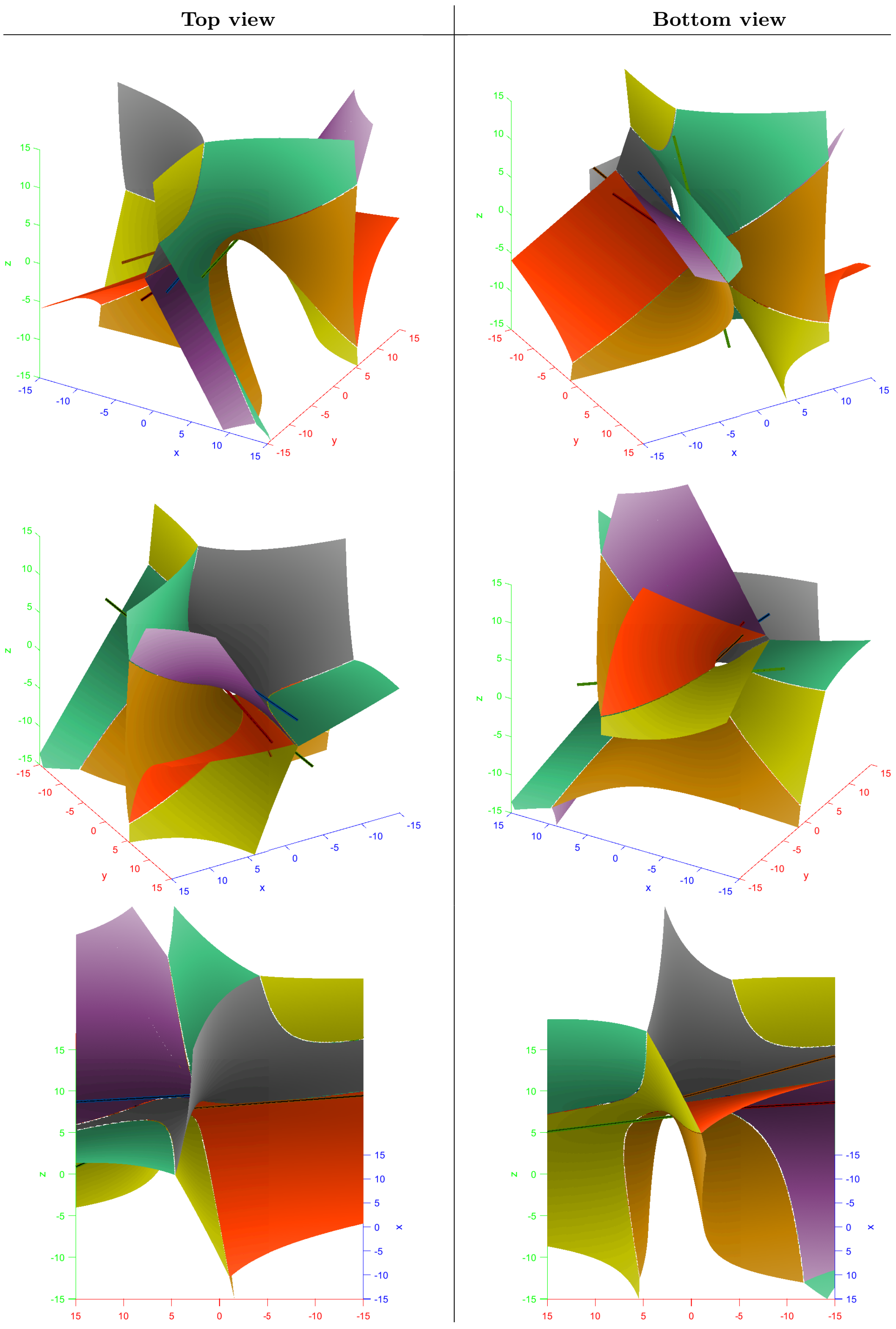


Figure 6 A NVD without vertex. The cell of the red line is the triangular prism.

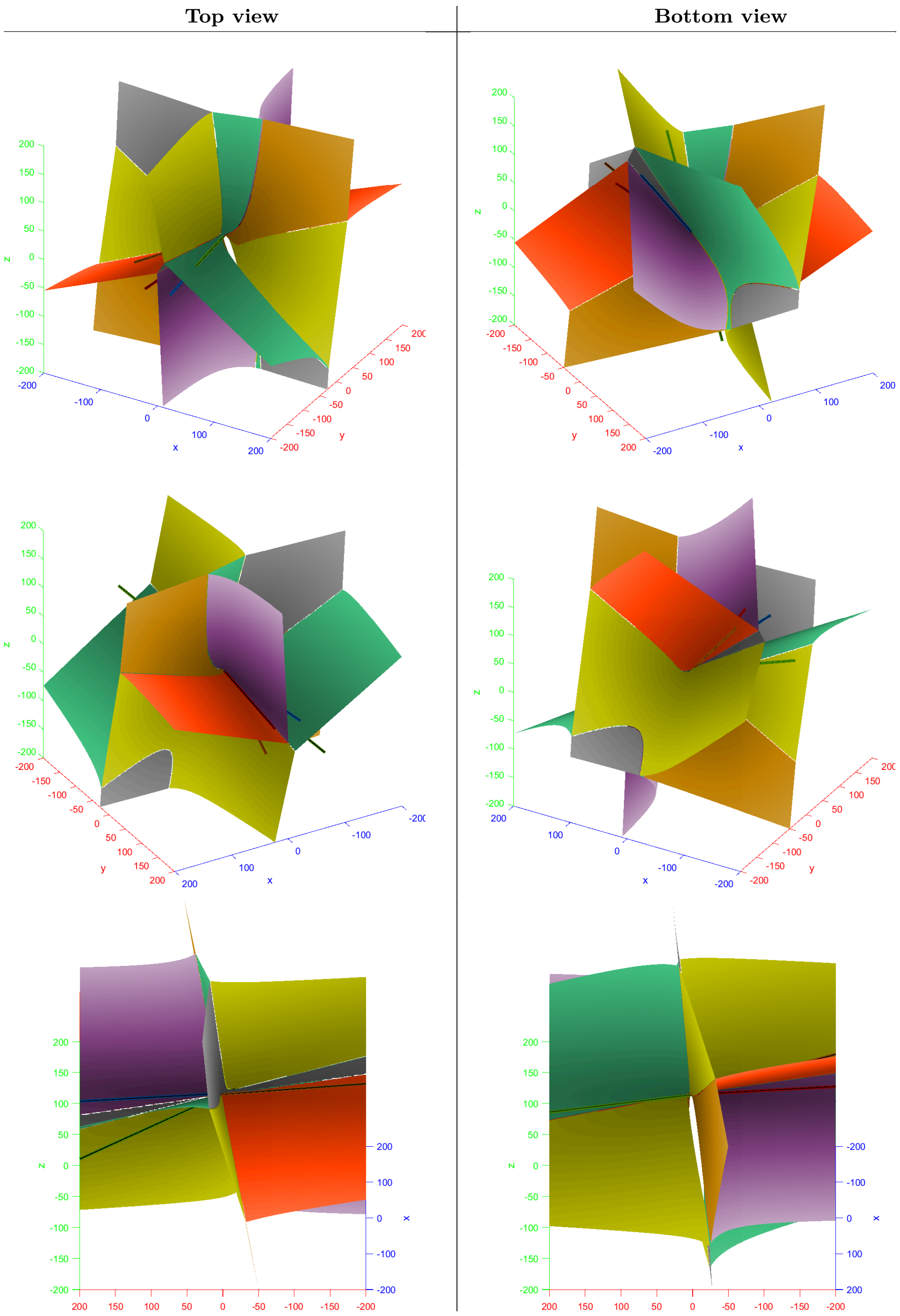


Figure 7 The same NVD as in Fig. 6 but more zoomed out to include all features.

References

1 Franz Aurenhammer, Bert Jüttler, and Günter Paulini. Voronoi diagrams for parallel halflines and line segments in space. In *28th International Symposium on Algorithms and Computation (ISAAC 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

2 Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi diagrams and Delaunay triangulations*. World Scientific Publishing Company, 2013.

3 Gill Barequet, Evanthia Papadopoulou, and Martin Suderland. Unbounded regions of high-order Voronoi diagrams of lines and line segments in higher dimensions. *Discrete & Computational Geometry*, pages 1–29, 2023.

4 Jean-Daniel Boissonnat, Micha Sharir, Boaz Tagansky, and Mariette Yvinec. Voronoi diagrams in higher dimensions under certain polyhedral distance functions. In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 79–88, 1995.

5 Bernard Chazelle. An optimal convex hull algorithm and new results on cuttings (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico*, pages 29–38. IEEE Computer Society, 1991.

6 Herbert Edelsbrunner and Raimund Seidel. Voronoi diagrams and arrangements. *Discrete & Computational Geometry*, 1:25–44, 1986.

7 Hazel Everett, Daniel Lazard, Sylvain Lazard, and Mohab Safey El Din. The Voronoi diagram of three lines. *Discrete & Computational Geometry*, 42(1):94–130, 2009.

8 Michael Hemmer, Ophir Setter, and Dan Halperin. Constructing the exact Voronoi diagram of arbitrary lines in three-dimensional space: with fast point-location. In *European Symposium on Algorithms*, pages 398–409. Springer, 2010.

9 Christian Icking and Lihong Ha. A tight bound for the complexity of Voroni diagrams under polyhedral convex distance functions in 3d. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 316–321, 2001.

10 Victor Klee. On the complexity of d-dimensional Voronoi diagrams. *Archiv der Mathematik*, 34(1):75–80, 1980.

11 Vladlen Koltun and Micha Sharir. Three dimensional Euclidean Voronoi diagrams of lines with a fixed number of orientations. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 217–226, 2002.

12 Joseph S. B. Mitchell and Joseph O'Rourke. Computational geometry column 42. *International Journal of Computational Geometry & Applications*, 11(5):573–582, 2001.

13 Raimund Seidel. On the number of faces in higher-dimensional Voronoi diagrams. In *Proceedings of the third annual symposium on Computational geometry*, pages 181–185, 1987.

Construction of 1-planar unit distance graphs with more edges than matchstick graphs[*]

Eliška Červenková[1]

1 **Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic**
`cervenel@seznam.cz`

Abstract

A matchstick graph is a plane graph with edges drawn as line segments of length 1. For this graph class, it was conjectured by Harborth that the maximum number of edges on n vertices is $\left\lfloor 3n - \sqrt{12n-3} \right\rfloor$. This was later proven by Lavollée and Swanepoel. In this paper we consider 1-planar unit distance graphs which are graphs that can be drawn in the plane such that every edge is drawn as a unit distance line segment and every edge is involved in at most 1 crossing. For these graphs, the best construction to achieve as many edges as possible was so far the construction for matchstick graphs. We present a construction of 1-planar unit distance graphs that have more edges than matchstick graphs on the same number of vertices. This answers an open problem of Gehér and Tóth.

1 Introduction

A *matchstick graph* is a graph that can be drawn in the plane with no crossings such that every edge is drawn as a line segment of length 1. This graph class was introduced by Harborth [3] [4] who proved that if the unit distance is also the smallest distance among the vertices, then the maximum number of edges of a matchstick graph on n vertices is $\left\lfloor 3n - \sqrt{12n-3} \right\rfloor$. He also showed that for each $n \in \mathbb{N}$, there are graphs on n vertices on triangular lattice which have exactly $\left\lfloor 3n - \sqrt{12n-3} \right\rfloor$ edges [2] [4]. We will denote the maximum number of edges of a matchstick graph on n vertices by $u_0(n)$. Later Lavollée and Swanepoel [5] proved that $\left\lfloor 3n - \sqrt{12n-3} \right\rfloor = u_0(n)$ holds for every $n \in \mathbb{N}$.

It is well known that a planar graph on $n \geq 3$ vertices can have at most $3n-6$ edges. If we allow that every edge can cross at most 1 other edge (we call this condition $1-crossings$), then the maximum number of edges on n vertices equals to $4n-8$ for $n \geq 4$ [6] [7]. Those graphs are called 1-*planar graphs*. We see that allowing 1-crossings led to bigger maximum number of edges which differs from the maximum number of edges of planar graphs in linear term.

A similar class of graphs is 1-*planar unit distance graphs* which are graphs that can be drawn in the plane such that every edge is drawn as a unit distance line segment and every edge is involved in at most 1 crossing. We will denote by $u_1(n)$ the maximum number of edges of such a graph on n vertices. In 2023 Gehér and Tóth [1] proved that $u_1(n) \leq 3n - \frac{\sqrt[4]{n}}{10}$, which is the best known upper bound. What is more interesting is the lower bound. Obviously it holds that $u_1(n) \geq u_0(n)$. Compared to planar graphs and 1-planar graphs we see that allowing 1-crossings does not help to get significantly more edges on n vertices as both lower and upper bounds have the same linear term. However, there have not been any better lower bounds. This leads to a question whether $u_0(n) = u_1(n)$ for every $n \in \mathbb{N}$ which was explicitly

[*] The research was supported by Czech Science Foundation research grant GAČR 23-04949X.

asked as an open problem in [1]. In this paper we show that it is not true for infinitely many specific $n \in \mathbb{N}$, and we present a construction of such graphs, which answers an open problem of Gehér and Tóth [1].

2 Main Result

▶ **Theorem 2.1.** *There are infinitely many $n \in \mathbb{N}$ for which $u_0(n) < u_1(n)$.*

Proof. We show a construction of graphs on $n = 3k^2 + 25k + 46$, $k \in \mathbb{N}_0$ vertices for which the required inequality holds. Let us start with the base case $n = 46$ and denote by G_0 the graph shown in Figure 1. It has 115 edges, whereas the maximum number of edges of matchstick graph on 46 vertices is 114.

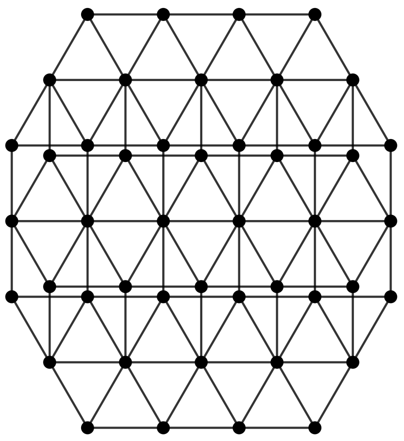


Figure 1 Graph G_0

This graph consists of copies of F - a graph on 6 vertices with 9 edges. The vertices can be represented by points in the plane with coordinates $v_1 = [0,0], v_2 = [0,1], v_3 = [1,0], v_4 = [1,1], v_5 = [\frac{1}{2}, \frac{\sqrt{3}}{2}]$ and $v_6 = [\frac{1}{2}, 1 + \frac{\sqrt{3}}{2}]$. The edges of F are then the unit distance line segments $v_1v_2, v_2v_4, v_4v_3, v_3v_1, v_1v_5, v_2v_5, v_3v_6, v_4v_6$ and v_5v_6. The only edges which are crossing are v_3v_4 and v_5v_6, so F is a 1-planar unit distance graph. From now on we consider the drawing shown in Figure 2 to be fixed.

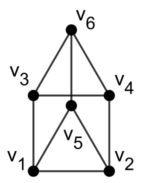


Figure 2 Graph F

To construct G_0 we at first use 5 copies of F denoted by $F^{(i)}$ for $i \in \{1, \ldots, 5\}$ with corresponding sets of vertices $\{v_j^{(i)}\}$ for $j \in \{1, \ldots, 6\}$ to construct a graph F'. For those copies of F in F', it holds that $v_2^{(t)} = v_1^{(t+1)}$ and $v_4^{(t)} = v_3^{(t+1)}$ for $t \in \{1, \ldots, 4\}$. We also add edges $v_5^{(t)}v_5^{(t+1)}$ and $v_6^{(t)}v_6^{(t+1)}$ for $t \in \{1, \ldots, 4\}$. All of them are represented by unit distance line segments. This procedure is shown in Figure 3. As all edges of F' are represented by unit distance line segments and we added 4 new crossings between edges $v_2^{(t)}v_4^{(t)}$ and $v_5^{(t)}v_5^{(t+1)}$, F' is also a 1-planar unit distance graph.

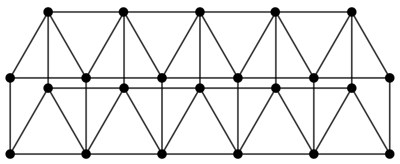


Figure 3 Graph F'

For the last step we define a *path of n triangles* denoted by T_n which is a graph on $2n+1$ vertices and $4n-1$ edges isometric to the set of points $\{[0+l,0],[\frac{1}{2}+m,\frac{\sqrt{3}}{2}:l\in\{0,\dots,n\},m\in\{0,\dots,n-1\}]\}$ where two points are connected if their distance is exactly 1. An example is shown in Figure 4. We denote by F'' a graph F' with added T_4 which is shown in Figure 4.

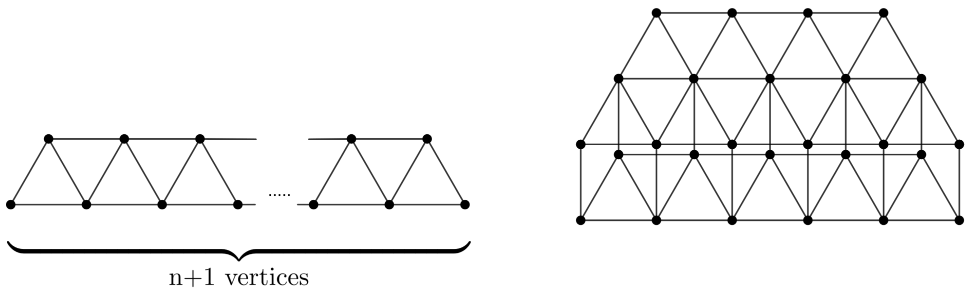


Figure 4 Path of n triangles T_n and a graph F''

The graph G_0 is then obtained by adding a reflected copy of F'' by the horizontal axis. By doing this construction we did not add any new crossings, except for the reflected images of the current crossings. As all edges are represented by unit distance line segments, G_0 is a 1-planar unit distance graph.

Then we proceed by induction. Our aim in each step is to construct a graph with its perimeter in an octagonal shape with lengths of its sides shown in Figure 5.

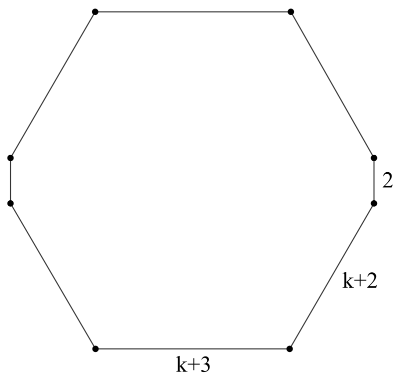


Figure 5 Lengths of the sides of the octagonal perimeter of G_k

To illustrate the construction we show in the following figures just the induction step from G_0 to G_1. The newly added vertices and edges are displayed in red.

Suppose we already have G_{k-1}. Then G_k is created in the following way:

(i) We denote by H a graph consisting of F and its reflected image along the horizontal axis together with 4 horizontal edges, as shown in Figure 6 below. Add 2 copies of H to G_{k-1}. The addition is also shown in Figure 6.

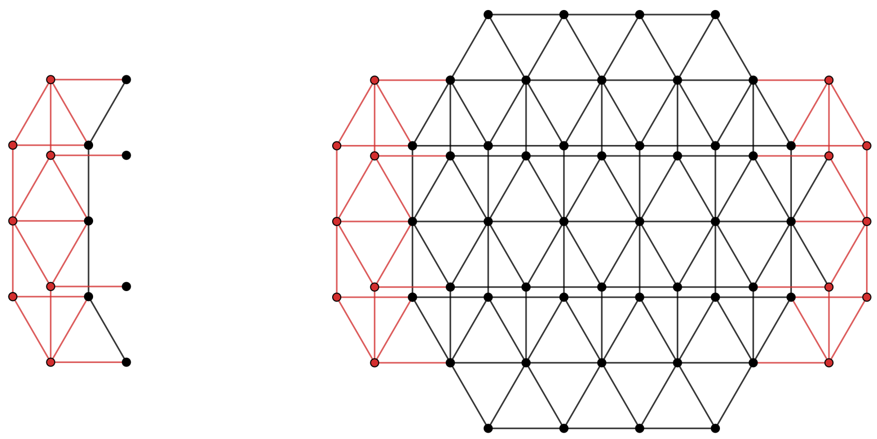


Figure 6 Graph H and two copies of H added to G_0

In total we added 14 vertices and 38 edges.

(ii) The lengths of the oblique sides of the octagonal perimeter of G_{k-1} are $k+1$. Hence add a T_k to each of the 4 "corners" of modified G_0 as shown in Figure 7, such that the side of T_k of length k coincides with the oblique side of G_{k-1}. Also add an edge between each T_k and H.

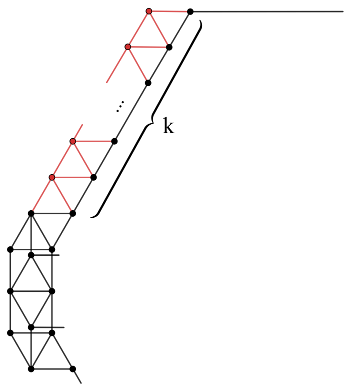


Figure 7 Adding vertices to the "corners" of modified G_{k-1} in general case

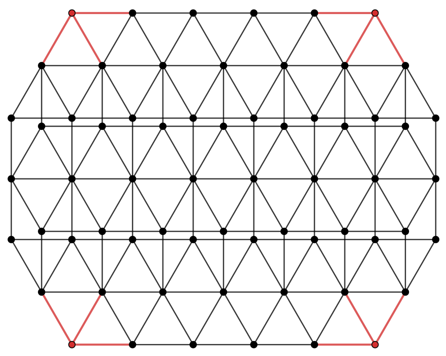


Figure 8 Adding vertices to the "corners" of modified G_0

In total we added $4k$ vertices and $12k$ edges.

(iii) Add T_{k+4} to the already created octagon from step (ii) such that the side of T_{k+4} of length $k+4$ coincides with horizontal sides of that octagon, which are of length $k+4$ too.

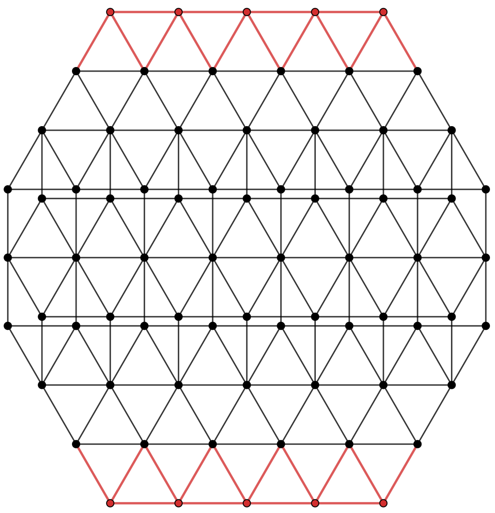


Figure 9 Adding two paths of triangles

In total we added $2(k+4)$ vertices and $2(3(k+4)-1)$ edges.

We can see that we can go from G_{k-1} to G_k by adding one "layer" around G_{k-1} consisting of $6k+22$ vertices and $18k+60$ edges. Equivalently, G_k is created by adding k "layers" to G_0. This means, that the number of vertices of G_k is equal to

$$46 + \sum_{i=1}^{k}(6i+22) = 46 + 22k + 6\frac{k(k+1)}{2} = 3k^2 + 25k + 46. \tag{1}$$

By the same argument, the number of edges of G_k is equal to

$$115 + \sum_{i=1}^{k}(18i+60) = 115 + 60k + 18\frac{k(k+1)}{2} = 9k^2 + 69k + 115. \tag{2}$$

Now by the results in [5] we know, that

$$u_0(n) = \left\lfloor 3n - \sqrt{12n-3} \right\rfloor. \tag{3}$$

For $n = 3k^2 + 25k + 46$ we get

$$u_0(3k^2+25k+46) = \left\lfloor 9k^2 + 75k + 138 - \sqrt{36k^2+300k+549} \right\rfloor, \tag{4}$$

and therefore

$$\begin{aligned} &u_1(3k^2+25k+46) - u_0(3k^2+25k+46) \geq |E(G_k)| - u_0(3k^2+25k+46) = \\ &= 9k^2+69k+115 - \left(\left\lfloor 9k^2+75k+138-\sqrt{36k^2+300k+549}\right\rfloor\right) > \\ &> 9k^2+69k+115 - ((9k^2+75k+138-\sqrt{36k^2+300k+549})+1) = \\ &= \sqrt{36k^2+300k+549} - 6k - 24 = \sqrt{(6k+24)^2+12k-27} - (6k+24). \end{aligned} \tag{5}$$

We see that for $k \geq 3$, we get

$$u_1(3k^2 + 25k + 46) - u_0(3k^2 + 25k + 46) > 0. \tag{6}$$

And a direct computation for $k \in \{0, 1, 2\}$ shows that for $n = 3k^2 + 25k + 46$, the inequality $u_1(n) > u_0(n)$ holds true for all $k \in \mathbb{N}_0$:

$$\begin{aligned} u_1(46) - u_0(46) &\geq |E(G_k)| - u_0(46) = 115 - 114 > 0, \\ u_1(74) - u_0(74) &\geq |E(G_k)| - u_0(74) = 193 - 192 > 0, \\ u_1(108) - u_0(108) &\geq |E(G_k)| - u_0(108) = 289 - 288 > 0. \end{aligned} \tag{7}$$

◀

The construction of G_k shows that allowing at most 1 crossing on every edge helps to build graphs with more edges on n vertices for infinitely many $n \in \mathbb{N}$. Let us have a look at $|E(G_k)|$ for first few k.

k	n=$\lvert V(G_k)\rvert$	$\lvert E(G_k)\rvert$	$u_0(n)$
0	46	115	114
1	74	193	192
2	108	289	288
3	148	403	401
4	194	535	533
5	246	685	683
10	596	1705	1703
50	8796	26065	26063

Table 1 Comparison of $u_0(n)$ and the number of edges of G_k

We see that the difference between $|E(G_k)|$ and $u_0(n)$ gets bigger for increasing k. This observation leads us to a question how much better our construction is compared to the best known construction for $u_0(n)$. However, a short analysis shows that we never get better result than $|E(G_k)| - u_0(n) = 2$.

▶ **Theorem 2.2.** *For $k \in \mathbb{N}_0$, it holds that*

$$\lim_{k \to \infty} |E(G_k)| - u_0(3k^2 + 25k + 46) = 2.$$

Proof. From (5) we have that

$$|E(G_k)| - u_0(3k^2 + 25k + 46) > \sqrt{36k^2 + 300k + 549} - (6k + 24), \tag{8}$$

and therefore

$$\lim_{k \to \infty} |E(G_k)| - u_0(3k^2 + 25k + 46) > \lim_{k \to \infty} \sqrt{36k^2 + 300k + 549} - (6k + 24) = \tag{9}$$

$$= \lim_{k \to \infty} \frac{12k - 27}{\sqrt{36k^2 + 300k + 549} + (6k + 24)} = 1. \tag{10}$$

Let us now estimate the upper bound:

$$\begin{aligned}
&|E(G_k)| - u_0(3k^2+25k+46) = \\
&= 9k^2+69k+115 - \left(\left\lfloor 9k^2+75k+138-\sqrt{36k^2+300k+549}\right\rfloor\right) < \\
&< 9k^2+69k+115-(9k^2+75k+138-\sqrt{36k^2+300k+549}-1) = \\
&= \sqrt{36k^2+300k+549}-(6k+22),
\end{aligned} \tag{11}$$

hence

$$\begin{aligned}
\lim_{k\to\infty} |E(G_k)| - u_0(3k^2+25k+46) &< \lim_{k\to\infty} \sqrt{36k^2+300k+549}-(6k+22) = \\
&= \lim_{k\to\infty} \frac{36k+65}{\sqrt{36k^2+300k+549}+(6k+22)} = 3.
\end{aligned} \tag{12}$$

As $\{|E(G_k)| - u_0(3k^2+25k+46)\}_{k=0}^{\infty}$ is a sequence of natural numbers, its limit, for which we computed

$$1 < \lim_{k\to\infty} |E(G_k)| - u_0(3k^2+25k+46) < 3, \tag{13}$$

is also a natural number. Therefore, $\lim_{k\to\infty} |E(G_k)| - u_0(3k^2+25k+46) = 2$. ◀

3 Conclusion

We showed that for infinitely many $n \in \mathbb{N}$, it holds that $u_1(n) > u_0(n)$. However, this construction allows us only to add at most 2 more edges between n vertices than the construction for $u_0(n)$. The question is whether it is the best construction for $u_1(n)$.

Problem 1: *Is* $\limsup_{n\to\infty} u_1(n) - u_0(n) = \infty$*?*

Our construction gives a higher number of edges than for matchstick graphs only for very specific numbers of vertices. Hence we ask if the gaps between them are really needed.

Problem 2: *Is it true that* $u_0(n) < u_1(n)$ *for every sufficiently large* $n \in \mathbb{N}$*?*

4 Note Added in Proof

After submitting the paper to EuroCG'25, we were able to answer Problem 1. The following is true:

▶ **Theorem 4.1.** *For every integer* t*, there exists a graph* G_t *with* $\mathcal{O}(t^2)$ *vertices such that*

$$|E(G_t)| - u_o(n) \geq t.$$

References

1 Panna Gehér and Géza Tóth. 1-Planar Unit Distance Graphs. In Stefan Felsner and Karsten Klein, editors, *32nd International Symposium on Graph Drawing and Network Visualization (GD 2024)*, volume 320 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:9, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.GD.2024.6`, `doi:10.4230/LIPIcs.GD.2024.6`.

2 Heiko Harborth. Lösung zu problem 664a. *Elem. Math.*, 29:14–15, 01 1974.

3 Heiko Harborth. Point sets with equal numbers of unit-distance neighbors (abstract). In *Discrete Geometry*, Oberwolfach, Tagungsbericht 31/1981, pages 11–12, Oberwolfach, 12–18 July 1981. Mathematisches Forschungsinstitut Oberwolfach.

4 Heiko Harborth. Match sticks in the plane. In The Lighter Side of Mathematics, editor, *Proceedings of the Eugène Strens Memorial Conference on Recreational Mathematics and its History held at the University of Calgary*, pages 281–28, Calgary, Alberta, August 1994. Mathematical Association of America, Washington, D.C.

5 Jérémy Lavollée and Konrad Swanepoel. A tight bound for the number of edges of matchstick graphs, 2023. URL: `https://arxiv.org/abs/2209.09800`, `arXiv:2209.09800`.

6 János Pach, Farhad Shahrokhi, and Mario Szegedy. Applications of the crossing number. In *Proceedings of the Tenth Annual Symposium on Computational Geometry*, SCG '94, page 198–202, New York, NY, USA, 1994. Association for Computing Machinery. `doi:10.1145/177424.177629`.

7 János Pach and Géza Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17:427–439, 1996. URL: `https://api.semanticscholar.org/CorpusID:20480170`.

On Local and Global Crossing Numbers[*]

Stefan Felsner[1], Michael Hoffmann[2], Tomas Hruz[2], Tillmann Miltzow[†3], Hannah Rotzoll[2], Shengzhe Wang[2], and Emo Welzl[2]

1 **Institut für Mathematik, Technische Universität Berlin, Germany**
felsner@math.tu-berlin.de

2 **Department of Computer Science, ETH Zürich, Switzerland**
{hoffmann,tomas.hruz,emo}@inf.ethz.ch, {hrotzoll,wangshe}@student.ethz.ch

3 **Department of Information and Computing Sciences, Utrecht University, The Netherlands**
t.miltzow@uu.nl

Abstract

A graph is *k-planar* if it can be drawn in $\mathbb{R}^2$ such that every edge has at most k crossings; such a drawing is called *k-plane*. The *local crossing number* $\operatorname{lcr}(G)$ of a graph G is the smallest $k \geq 0$ such that G is k-planar. The (global) *crossing number* $\operatorname{cr}(G)$ of a graph G is the minimum number of crossings over all drawings of G. Counting the number of crossings per edge in any $\operatorname{lcr}(G)$-plane drawing of G yields $2 \cdot \operatorname{cr}(G) \leq m \cdot \operatorname{lcr}(G)$, where m denotes the number of edges in G. We are interested in graphs for which this inequality holds with equality. As a main result we show that such graphs exist. More precisely, for any given $k \in \mathbb{N}$, we construct an infinite family of graphs that admit a minimum-crossing drawing in which every edge is crossed exactly k times.

1 Introduction

Are there drawings of graphs where every edge is crossed? Indeed, such drawings of K_n were first mentioned by Ringel [17] and later systematically studied by Harborth and Mengersen [11]. However, their drawings have many "unnecessary" crossings. So, what if we restrict to *minimum-crossing* drawings, that is, drawings with a minimum number $\operatorname{cr}(G)$ of edge crossings among all drawings of a graph G? Surprisingly, the answer remains the same: Ábrego, Aichholzer, Fernández-Merchant, Ramos, and Vogtenhuber [1] describe a family of drawings of K_n, for odd $n \geq 11$, where every edge is crossed and the total number of crossings is

$$H(n) = \frac{1}{4}\left\lfloor\frac{n}{2}\right\rfloor\left\lfloor\frac{n-1}{2}\right\rfloor\left\lfloor\frac{n-2}{2}\right\rfloor\left\lfloor\frac{n-3}{2}\right\rfloor .$$

According to the famous (Guy-Harary-)Hill Conjecture [10], we have $H(n) = \operatorname{cr}(K_n)$ and thus the aforementioned drawings are minimum-crossing. Despite supporting evidence (e.g., [4]), the conjecture is proven for $n \leq 14$ only [3]. Still, this leaves us with minimum-crossing drawings of K_{11} and K_{13} in which every edge is crossed. Taking any number of disjoint copies of these graphs and drawings, we obtain an infinite family of graphs with minimum-crossing drawings where every edge is crossed. In this work, we study the problem under the lens of k-planarity and give a general answer for any number k of crossings.

[*] This research was initiated at the 21st Gremo's Workshop on Open Problems (GWOP 2024), Pura (TI), Switzerland, June 3–7, 2024.
[†] supported by the NWO under project no. VI.Vidi.213.150.

▶ **Theorem 1.** *For every $k, n_0 \in \mathbb{N}$ there exists a graph G on $n \geq n_0$ vertices such that G has a drawing with $\text{cr}(G)$ crossings, in which every edge is crossed exactly k times.*

In particular, in every k-plane minimum-crossing drawing of G every edge has exactly k crossings. Our motivation is to shed some light onto the relationship between the local and the (global) crossing number of graphs. To explain this relationship we introduce some standard terminology. A *drawing* of a graph $G = (V, E)$ is a map $\gamma : G \to \mathbb{R}^2$ that maps each vertex $v \in V$ to a point $\gamma(v) \in \mathbb{R}^2$ and each each edge $uv \in E$ to a simple (injective) curve $\gamma(uv)$ with endpoints $\gamma(u)$ and $\gamma(v)$, such that: (1) γ is injective on V; (2) $\gamma(uv) \cap \gamma(V) = \{\gamma(u), \gamma(v)\}$, for all $uv \in E$; and (3) for all $e_0, e_1 \in E$ with $e_0 \neq e_1$, the curves $\gamma(e_0)$ and $\gamma(e_1)$ have at most finitely many intersections, and each such intersection is either a common endpoint or a proper, transversal *crossing* between exactly two edges. A drawing is *simple* if every pair of edges has at most one common point (which is either a crossing or a common endpoint). The zoo of crossing numbers [18] is a very lively part of combinatorial geometry and graph drawing. The *crossing number* $\text{cr}(D)$ of a drawing D is the number of edge crossings in D. The *crossing number* $\text{cr}(G)$ of a graph G is the minimum of $\text{cr}(D)$ over all drawings D of G. The *local crossing number* $\text{lcr}(D)$ of a drawing D is the maximum number of crossings on any edge in D. The *local crossing number* $\text{lcr}(G)$ of a graph G is the minimum of $\text{lcr}(D)$ over all drawings D of G. A graph G is *k-planar* if $\text{lcr}(G) \leq k$. The *k-planar crossing number* $\text{cr}_{k\text{-pl}}(G)$ is the minimum of $\text{cr}(D)$ over all drawings D of G with $\text{lcr}(D) \leq k$; or $\text{cr}_{k\text{-pl}}(G) = \infty$ if no such drawing exists. By definition, for every graph G we have

$$\text{cr}(G) \leq \text{cr}_{k\text{-pl}}(G). \tag{I1}$$

Further, as every graph G admits a drawing D with $\text{lcr}(D) = \text{lcr}(G)$, we have

$$\text{cr}(G) \leq \text{cr}(D) \leq \frac{m\text{lcr}(D)}{2} = \frac{m}{2}\text{lcr}(G)\,, \tag{I2}$$

where m denotes the number of edges in G. It is natural to wonder if and for what graphs these two inequalities are tight. Theorem 1 answers the if-part of this question by providing an infinite family of graphs for which both inequalities are tight. Chimani, Kindermann, Montecchiani and Valtr [7, 8] explored the opposite end of the spectrum for (I1) and constructed a family of 1-planar graphs G with $\text{cr}_{1\text{-pl}}(G) \in \Omega(n)$ and $\text{cr}(G) = 2$, where n denotes the number of vertices in G. Similar results have also been obtained for other beyond-planar graph classes [6, 20]. The maximum number of edges in a simple k-planar graph on n vertices is known to be at most $c_k(n-2)$, where $c_0 = 3$, $c_1 = 4$ [5], $c_2 = 5$ [15, 16], $c_3 = 5.5$ [13, 14], $c_4 = 6$ [2], and $c_k \leq 3.81\sqrt{k}$, for general $k \geq 5$ [2]. So we could plug these bounds into (I2) to obtain an upper bound on $\text{cr}(G)$ in terms of n, the number of vertices in G. However, the graphs that maximize m might be quite different from the graphs for which (I2) is tight. For instance, for a 1-planar graph G it is known (see, e.g. [19, Proposition 4.4]) that it admits a drawing D such that

$$\text{cr}(G) \leq \text{cr}_{1\text{-pl}}(G) \leq \text{cr}_{1\text{-pl}}(D) \leq n-2\,, \tag{I3}$$

which beats the bound we get by plugging $m \leq 4n$ into (I2) by a factor of two. The bounds in (I2) and (I3) are somewhat conflicting. In fact, we show that if a graph G admits a drawing D for which both (I2) and (I3) are tight, then G is planar and thus $\text{cr}(G) \ll \text{cr}(D)$.

▶ **Theorem 2.** *Let D be a simple drawing of a graph G on n vertices where every edge is crossed exactly once. Then: (1) If $\text{cr}(D) = n-2$, then G is planar and disconnected; (2) If $\text{cr}(D) = n-3$, then G is planar.*

2 Preliminaries and Outline

Before we delve into the proof of Theorem 1, let us note that the statement for $k = 1$ can be obtained using edge subdivisions.

▶ **Lemma 3.** *The following statements are equivalent: (1) There exists a minimum-crossing drawing of a graph in which every edge is crossed; (2) There exists a minimum-crossing drawing of a graph in which every edge is crossed exactly once.*

Proof. The implication (2)⇒(1) is trivial. So it remains to prove (1)⇒(2). Let D be a minimum-crossing drawing of a graph G in which every edge is crossed. For every edge e in D and every pair c, c' of consecutive crossings along e in D we insert a new vertex between c and c' on e and also subdivide the corresponding edge in G. As a result, we obtain a graph G' and a drawing D' of G' such that G' is a subdivision of G and D' is a subdivision of D. By construction every edge in D' is crossed exactly once and $\mathrm{cr}(D) = \mathrm{cr}(D')$. As edge subdivisions have no effect on the crossing number of a graph, we have $\mathrm{cr}(G') = \mathrm{cr}(G) = \mathrm{cr}(D) = \mathrm{cr}(D')$ and thus D' is a minimum-crossing drawing of G'. ◀

The above subdivision argument also appears in the proof of Theorem 1. Our main challenge is to certify that a given drawing is minimum-crossing, which is very difficult in general. To get around this, we employ edge weights and correspondingly weighted crossing numbers as a framework to guarantee that the minimum-crossing drawing of the graph at hand is essentially unique. Our construction uses a highly symmetric graph based on the icosahedron graph. The symmetries of this graph help to simplify the argument. With a proper choice of weight assignment and subdivision, we can transfer the weighted minimum-crossing drawing into our target drawing where every edge is crossed exactly k times.

3 Minimum-crossing drawings in which every edge is crossed k times

We use the icosahedron graph $\mathsf{Ico} = (X, E_I)$ as a base graph, which is a plane triangulation with 12 vertices and 20 faces. It admits a combinatorially unique plane drawing, depicted by red edges in Figure 1.

Let $T = (V, E)$ be a plane triangulation. For an edge e of T, the two triangles incident to e form a 4-gon $Q(e)$. One of the diagonals of $Q(e)$ is e, we call the other diagonal of $Q(e)$ the *p-dual edge* of e. The *double-icosahedron* Dco is obtained from the icosahedron graph $\mathsf{Ico} = (X, E_I)$ by adding the p-dual edge of every edge $e \in E_I$, that is, we have $\mathsf{Dco} = (X, E_I \cup E_P)$, where E_P denotes the set of edges that are p-dual to some edge in E_I. It is interesting to note that the graph (X, E_P) is again an icosahedron graph, and Dco is a 10-regular graph on 12 vertices, isomorphic to a K_{12} with a perfect matching removed.

If we take a closer look into the drawing shown in Figure 1, we can easily count the crossing number of it since each of the 30 edges of E_I is only crossed by its p-dual edge and each edge of E_P is crossed by five other edges. Consequently, there is a total of $\frac{1}{2}(30+5\cdot30) = 90$ crossings in the drawing. We further remark that this matches the conjecture presented by Mohar [12] for K_n^t, which denotes the graph obtained from K_n by removing a matching of size t. Though an attempt to compute the exact crossing number of Dco by a computer[9] failed due to its inherent complexity, we conjecture $\mathrm{cr}(\mathsf{Dco}) = 90$ as well.

Weighted Minimum-Crossing Drawing. To derive our results, we first consider a weighted variant of the crossing number. Let $w : E \to \mathbb{R}^+$ be a weight assignment on the edges of a graph G. The *weighted crossing number* $\mathrm{cr}_w(D)$ of a drawing D of G is the sum of the

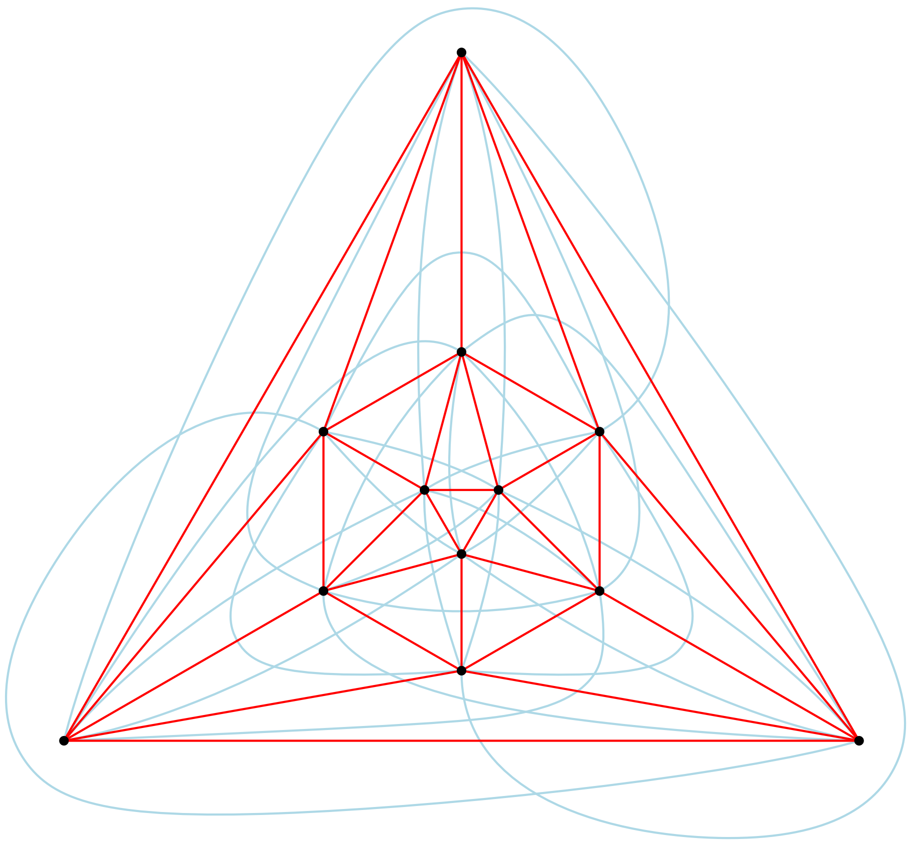


Figure 1 A drawing of Dco where every edge is crossed.

product $w(e) \cdot w(f)$ taken over all edge crossings $e \cap f$ in D. The weighted crossing number $\mathrm{cr}_w(G)$ of a graph G is the minimum of the weighted crossing numbers of its drawings.

▶ **Lemma 4.** *There is a weight assignment w on the edges of* Dco *such that the drawing shown in Figure 1 minimizes the weighted crossing number of this weighted graph.*

Proof. Consider any minimum-crossing drawing Γ of Dco. Recall that $\mathsf{Dco} = (X, E_I \cup E_P)$ with $|E_I| = |E_P| = 30$. If $w(e) = a$ for all $e \in E_I$ and $w(e) = 1$ for all $e \in E_P$, then for the drawing Γ_0 in Figure 1 we have $\mathrm{cr}_w(\Gamma_0) = 30a + 60$. We choose $a = 32$ s.t. $a^2 = 1024 > 30a + 60 = 1020$, and no pair of edges from E_I can cross in Γ. Hence, the drawing of (X, E_I) in Γ is a plane drawing Ico of the icosahedron. As every edge in E_P has to cross at least one edge of Ico, we have $\mathrm{cr}_w(\Gamma) \geq 30a$. At most one edge of E_P crosses two or more—and if so, exactly two—edges of Ico in Γ (else $\mathrm{cr}_w(\Gamma) \geq 30a + 2a = a^2 > \mathrm{cr}_w(\Gamma_0)$). Hence in at least 18 of the triangles of Ico, the three p-duals of the triangle edges cross inside the triangle. This yields at least $3 \cdot 18$ crossings between edges in E_P. Since $31a + 3 \cdot 18 > 30a + 60$, all the p-edges in Γ must be drawn in the union of two triangles and cross four other p-edges, just as in Γ_0. It follows that $\mathrm{cr}_w(\Gamma) = \mathrm{cr}_w(\Gamma_0)$. ◀

Let G be a graph and $w : E \to \mathbb{N}$ be an integral weight, we define the multigraph G_w by replacing each edge e of G by $w(e)$ copies of e, i.e., a set of $w(e)$ parallel edges connecting the vertices of e. Every drawing of G has a corresponding drawing of G_w by replacing the drawn edge e by a bundle of $w(e)$ edges in a small neighborhood of e. This construction shows that $\mathrm{cr}(G_w) \leq \mathrm{cr}_w(G)$. Given a drawing of G_w and an edge e of G we can look at all copies of e in the drawing and choose one, say e', involved in a minimum number of crossings. Now redraw the $w(e)$ edges of the bundle of e in a small neighborhood of e'. By the choice of e' this operation can only decrease the crossing number of the drawing. Repeating this for all edges of G we arrive at a drawing of G_w where parallel edges behave the same, i.e., which comes from a drawing of G. Therefore $\mathrm{cr}(G_w) = \mathrm{cr}_w(G)$.

▶ **Theorem 1.** *For every $k, n_0 \in \mathbb{N}$ there exists a graph G on $n \geq n_0$ vertices such that G has a drawing with $\mathrm{cr}(G)$ crossings, in which every edge is crossed exactly k times.*

Proof. We first deal with the case $k = 1$. Let Dco^+ be the multigraph obtained with multiplicity 64 for edges of E_I and multiplicity 2 for edges of E_P. The drawing Γ_0 By a similar analysis as shown in the proof of Lemma 4, this induces a minimum-crossing drawing of Dco^+ by replacing each red edge by a bundle of 64 parallel edges and each light blue edge by 2 parallel edges. For every edge e and every pair of consecutive crossings on e we place a new subdivision vertex on e; see Figure 2 (top). In this way, we obtain a subdivision Dco_s^+ of Dco^+ that has a minimum-crossing drawing where every edge is crossed exactly once. To obtain arbitrarily large graphs, we take sufficiently many disjoint copies of Dco.

For $k \geq 2$ we adapt the weighting w such that if D_w is a minimum-crossing drawing of Dco^+, then in D_w the number of crossings on each edge is a multiple of k. An appropriate weighting is obtained by using the weight $64k$ for edges in E_I and $2k$ for edges in E_P. The subdivision vertices of Dco_s^+ are placed on the edges of D_w such that every edge of Dco_s^+ has exactly k crossings. Figure 2 (bottom) illustrates the construction in the case $k = 2$. ◀

4 On Graphs for which Both Upper Bounds are Tight

Due to space constraints we present the proof of the first statement in Theorem 2 only; the proof for the second statement can be found in a later full version.

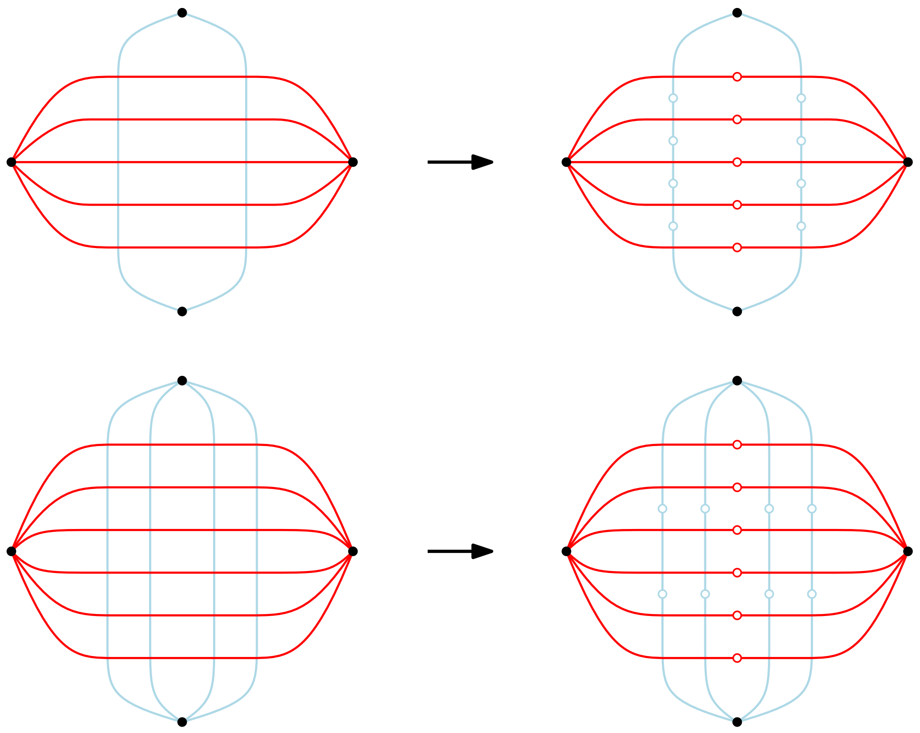


Figure 2 Subdividing edges to obtain a k-plane drawing, for $k = 1$ (top) and $k = 2$ (bottom).

▶ **Theorem 2.** *Let D be a simple drawing of a graph G on n vertices where every edge is crossed exactly once. Then: (1) If $\mathrm{cr}(D) = n - 2$, then G is planar and disconnected; (2) If $\mathrm{cr}(D) = n - 3$, then G is planar.*

Proof of (1). Let D be a drawing of G with $n - 2$ crossings such that every edge is crossed exactly once. Let D^+ be obtained from D by adding a maximal set of uncrossed edges, and let H be the plane drawing consisting of the new edges only. Each crossing pair of edges in D corresponds to a 4-face of H. Hence, H is a plane graph with n vertices and $n - 2$ many 4-faces. From Euler's formula it follows that H is a quadrangulation. In particular H is bipartite. Transferring the 2-coloring of H to D, we observe that for every pair e, e' of crossing edges the endpoints of e have the same color and both endpoints of e' have the other color. Hence, each of the two color classes induces a plane graph. Denote these two graphs by D_1 and D_2. As every edge in D is crossed, the graph D is the disjoint union of D_1 and D_2. Thus, we obtain a plane drawing of D by drawing D_1 and D_2 side by side. ◀

5 Open Problems

Our construction to prove Theorem 1 uses edge subdivisions extensively. Can a similar statement be obtained for graphs with minimum vertex degree three or higher? In Theorem 2 we show that there are no graphs for which both upper bounds (I1) and (I2) on the crossing number are (almost) tight. How far can we further relax the "almost" tightness condition?

References

1 Bernardo M. Ábrego, Oswin Aichholzer, Silvia Fernández-Merchant, Pedro Ramos, and Birgit Vogtenhuber. Non-shellable drawings of K_n with few crossings. In *Proc. 26th Canad. Conf. Comput. Geom.*, pages 7:1–7:6, 2014. URL: `https://www.cccg.ca/proceedings/2014/papers/paper07.pdf`.

2 Eyal Ackerman. On topological graphs with at most four crossings per edge. *Comput. Geom.*, 85:101574, 2019. `doi:10.1016/J.COMGEO.2019.101574`.

3 Oswin Aichholzer. Another small but long step for crossing numbers: cr(13) = 225 and cr(14) = 315. In *Proc. 33rd Canad. Conf. Comput. Geom.*, pages 72–77, 2021. URL: `https://projects.cs.dal.ca/cccg2021/wordpress/wp-content/uploads/2021/08/CCCG2021.pdf`.

4 József Balogh, Bernard Lidický, and Gelasio Salazar. Closing in on Hill's Conjecture. *SIAM J. Discrete Math.*, 33(3):1261–1276, 2019. `doi:10.1137/17M1158859`.

5 Rainer Bodendiek, Heinz Schumacher, and Klaus Wagner. Bemerkungen zu einem Sechsfarbenproblem von G. Ringel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 53:41–52, 1983. `doi:10.1007/BF02941309`.

6 Markus Chimani, Torben Donzelmann, Nick Kloster, Melissa Koch, Jan-Jakob Völlering, and Mirko H. Wagner. Crossing numbers of beyond planar graphs re-revisited: A framework approach. In *Proc. 32nd Internat. Sympos. Graph Drawing*, volume 320 of *Leibniz Internat. Proc. Informatics (LIPIcs)*, pages 33:1–33:17. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024. URL: `https://doi.org/10.4230/LIPIcs.GD.2024.33`.

7 Markus Chimani, Philipp Kindermann, Fabrizio Montecchiani, and Pavel Valtr. Crossing numbers of beyond-planar graphs. In *Proc. 27th Internat. Sympos. Graph Drawing*, volume 11904 of *Lecture Notes in Computer Science*, pages 78–86, 2019. `doi:10.1007/978-3-030-35802-0\_6`.

8 Markus Chimani, Philipp Kindermann, Fabrizio Montecchiani, and Pavel Valtr. Crossing numbers of beyond-planar graphs. *Theor. Comput. Sci.*, 898:44–49, 2022. `doi:10.1016/J.TCS.2021.10.016`.

9 Markus Chimani and Tilo Wiedera. An ILP-based proof system for the crossing number problem. In *24th Annual European Symposium on Algorithms*, volume 57 of *LIPIcs*, pages 29:1–29:13, 2016. `doi:10.4230/LIPICS.ESA.2016.29`.

10 Frank Harary and Anthony Hill. On the number of crossings in a complete graph. *Proc. Edinburgh Math. Soc.*, 13(4):333–338, 1963. `doi:10.1017/S0013091500025645`.

11 Heiko Harborth and Ingrid Mengersen. Edges without crossings in drawings of complete graphs. *J. Combin. Theory Ser. B*, 17(3):229–311, 1974. `doi:10.1016/0095-8956(74)90035-5`.

12 Bojan Mohar. On a conjecture by Anthony Hill, 2020. arXiv 2009.03418. URL: `https://arxiv.org/abs/2009.03418`, `arXiv:2009.03418`.

13 János Pach, Radoš Radoičić, Gábor Tardos, and Géza Tóth. Improving the crossing lemma by finding more crossings in sparse graphs. In *Proc. 20th Annu. Sympos. Comput. Geom.*, pages 68–75, 2004. `doi:10.1145/997817.997831`.

14 János Pach, Radoš Radoičić, Gábor Tardos, and Géza Tóth. Improving the crossing lemma by finding more crossings in sparse graphs. *Discrete Comput. Geom.*, 36(4):527–552, 2006. `doi:10.1007/s00454-006-1264-9`.

15 János Pach and Géza Tóth. Graphs drawn with few crossings per edge. In *Proc. 4th Internat. Sympos. Graph Drawing*, volume 1190 of *Lecture Notes in Computer Science*, pages 345–354, 1996. `doi:10.1007/3-540-62495-3\_59`.

16 János Pach and Géza Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17(3):427–439, 1997. `doi:10.1007/BF01215922`.

17 Gerhard Ringel. Extremal problems in the theory of graphs. In *Theory of Graphs and Its Applications, Proc. Sympos. Smolenice*, pages 85–90, 1963.

18 Marcus Schaefer. The graph crossing number and its variants: A survey. *Electron. J. Comb.*, Dynamic Surveys, DS21(8):1–166, 2024. doi:10.37236/2713.

19 Yusuke Suzuki. 1-planar graphs. In *Beyond Planar Graphs, Communications of NII Shonan Meetings*, pages 47–68. Springer, 2020. doi:10.1007/978-981-15-6533-5_4.

20 Nathan van Beusekom, Irene Parada, and Bettina Speckmann. Crossing numbers of beyond-planar graphs revisited. *J. Graph Algorithms Appl.*, 26(1):149–170, 2022. doi:10.7155/JGAA.00586.

Minimizing Vertical Length in Linked Bar Charts[*]

Steven van den Broek[1], **Marc van Kreveld**[2], **Wouter Meulemans**[†1], **and Arjen Simons**[‡1]

1 **TU Eindhoven, the Netherlands**
`{s.w.v.d.broek|w.meulemans|a.simons1}@tue.nl`
2 **Utrecht University, the Netherlands**
`m.j.vankreveld@uu.nl`

Abstract

We consider linked bar charts, in which each bar is partitioned into blocks, and blocks are linked between bars through orthogonal lines. Given a fixed bar order, we study the problem of minimizing the vertical length of these links by stacking the blocks appropriately within each bar. We describe algorithms that handle different assumptions on the links between bars, including for example the case of having bars sorted in order of height.

1 Introduction

In information visualization, it is often important to show the certainty of information. Given a collection of k sets $S_1, \ldots, S_k$, the size of each set can have a certain part and an uncertain part, where the uncertain part is independent of the uncertain parts of the other sets. But it could also have an uncertain component of, say, the set S_1, where we know that if it does not belong to S_1, it must belong to, say, S_2. In this case, there is a *dependent uncertainty* between the size of sets S_1 and S_2. This can model, for example, groups of voters that hesitate between two political parties for predicting election outcomes, or pollution of factories near country borders, where the actual country affected will be one of the two. In such cases, a *pairwise uncertainty* in size, with a dependence between S_1 and S_2 exists.

Van Beusekom et al. [5] introduce a visualization for these pairwise uncertainties using bar charts, where the pink *blocks* on top represent the pairwise uncertainty (see Fig. 1). Each pink block appears twice in the same size, once in each set to which it may belong, with a *link*, represented by a polyline, that connects the two blocks.

There are several optimization possibilities when reordering the bars horizontally and reordering the blocks vertically, such as the maximal horizontal or vertical span of the links, the total horizontal/vertical link length, the cut-width and the number of link crossings.

Related work. When optimizing for the horizontal span/length, number of link crossings or cut-width, the bars and their links can be represented as a one-page book embedding, where each bar represents a node and each link represents an edge. We know that for a one-page book embedding of a graph G it is NP-complete to minimize edge crossings [3] and that it can be drawn without edge crossings if and only if G is outerplanar [4]. Frederickson and Hambrusch [1] provide polynomial-time algorithms to minimize the total edge length, maximum edge length, the cutwidth, and the bandwidth when G is outerplanar. These problems are NP-complete for general graphs [2].

[*] Research on the topic of this paper was initiated at the 8th Workshop on Applied Geometric Algorithms (AGA 2024) in Otterlo, The Netherlands.
[†] W. Meulemans is partially supported by the Dutch Research Council (NWO) under project number VI.Vidi.223.137.
[‡] A. Simons is supported by the Dutch Research Council (NWO) under project number VI.Vidi.223.137.

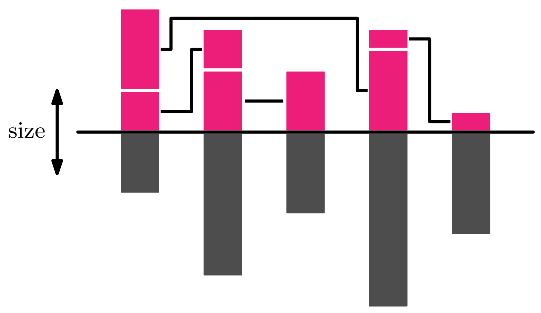


Figure 1 A linked bar chart [5] with pairwise uncertainty between the linked pink blocks. The certain size (gray) and independent uncertainty (not shown) are irrelevant to our problem.

Contributions and organization. We study the minimization of the vertical link length by reordering the blocks in each bar, for a given bar order. Our main contributions are polynomial-time algorithms for restricted variants, based on whether there are "dependent links", whose vertical length cannot be optimized by optimizing bars individually. In Section 2 we introduce our terminology and notation, along with general observations. Sections 3 through 5 present our algorithmic results.

2 Notation, definitions and observations

Our input is an edge-weighted graph $G = (V, E, w)$, where the V represents a sequence of n sets $S_1, \ldots, S_n$ in fixed order and $E \subseteq V^2$ is a set of m edges, of which each edge $(S, S') \in E$ represents a pairwise uncertainty between two sets $S, S' \in V$. The edge weights $w : E \to \mathbb{R}^+$ represent the *size* of these uncertainties.

The *span* of an edge is the subsequence of sets in V between its endpoints, including these endpoints. The *intermediate* sets of an edge refer to its span minus its endpoints. We call an edge e *contained* in another edge e', if the span of e is a subset of the span of e'.

Bars, blocks and links. To visualize the pairwise uncertainties in G, we draw the sets in V as a sequence of bars $\mathcal{B} = \{B_1, \ldots, B_n\}$, arranged on a horizontal baseline with a spacing of two units of width between the horizontal centers of adjacent bars. Note that the horizontal position of the bars is fixed by the order of $\mathcal{B}$. Each set S_i corresponds to bar B_i for all $1 \leq i \leq n$; we use bar to refer to the set in V as well as its visual representation, as there is a one-to-one mapping between the two.

An edge e induces a *block* in each of the two bars it connects: a rectangle of unit width and height $w(e)$; the width is not relevant to the optimization problems in this abstract, since we optimize vertical length. We refer to the height of b as $h : E \to \mathbb{R}^+$, meaning $h(b) = w(e)$.

We construct a bar B_i by stacking the blocks that correspond to the edges incident to S_i on top of each other; see Fig. 1. Therefore the height of a bar equals the sum of its block heights. We refer to the height of a bar as $h \colon E \cup V \to \mathbb{R}^+$.

We draw an edge e as an orthogonal *link* $\bar{e}$ that connects the two blocks corresponding to e; see also Fig. 1. A link connects the centers of two blocks; we refer to these centers as the endpoints of the link. If at least one endpoint is placed higher than all intermediate bars of e, then the link generally has two bends; a link may be horizontal in some cases and have zero bends. Otherwise, the link is drawn using four bends. Links always pass over all intermediate bars. We assume that between two consecutive bars, there is enough horizontal space for the links to run between those bars, and ignore this issue from now on.

Stacking blocks. Note that as the order of bars is fixed, the horizontal length of links is as well. We aim to minimize the total vertical length of the links by stacking the blocks that comprise a bar appropriately while avoiding crossings between links that connect to the same bar. A stacking of blocks corresponds to an order on the incident edges of a bar.

Consider a bar B_j. All incident edges $\{B_i, B_j\}$ to an earlier bar B_i go towards the left from B_j. We denote this ordered sequence of edges by $L_j = \{l_1, \ldots, \}$, sorted in decreasing order of index of the other bar. Similarly, the rightward edges are denoted by $R_j = \{r_1, \ldots, \}$—the edges $\{B_j, B_k\}$ with $k > j$—in increasing order of index of the other bar. To avoid crossings between links that originate from a common bar, the stacking order must contain both L_j and R_j as a subsequence. That is, the stacking order must be some merge of these two ordered sequences. Any merge is valid, but they incur different vertical link lengths. For ease of notation, we identify an edge e in L_j or R_j with the corresponding block in bar B_j.

Merging leftward and rightward subsequences in this way implies that the number of crossings in our visualization is equal to the number of crossings in the one-page book embedding of G that respects the fixed vertex order. Two links intersect if and only if their sets of intermediate bars have a non-empty intersection and neither is a subset of the other.

Consider some block b in a bar B_j for edge $l_i \in L_j$. Its vertical center point is determined by its height and all blocks prior to it in L_j—the order of the blocks in L_j is fixed after all—but also by all blocks in R_j that are chosen to be below it in the stacking order; denote by k the last block in R_j that is still before (below) b. Specifically, we find that $y = y(b, k) = \frac{1}{2}h(b) + \sum_{x=1}^{i-1} h(l_x) + \sum_{x=1}^{k} h(r_x)$. Hence, the center coordinate of a block is influenced only by the number of blocks in the other sequence occurring before it. The situation is symmetrical for a block in R_j.

Link types. Consider an edge $e = \{B_i, B_j\}$ with $i < j$. Its blocks b and b' must be in R_i and L_j. Let $\uparrow b = y(b, |L_i|)$ and $\downarrow b = y(b, 0)$ denote the highest and lowest possible y-coordinate for block b and analogously define $\uparrow b' = y(b', |R_j|)$ and $\downarrow b' = y(b', 0)$. Let $H = \max_{i<x<j} h(B_x)$ denote the height of the highest intermediate bar.

If $H \geq \uparrow b$ (and analogously if $H \geq \uparrow b'$), $\bar{e}$ must go up to H from b before continuing to b'; see Fig. 2. The vertical length is $|H - y| + |H - y'|$, for given center coordinates y for b and y' for b'. Therefore, both b and b' must be as close as possible to H, to minimize vertical length. Their quality in the stacking order for B_i and B_j is independent; we thus call the link *independent*. In either case, H is the *target* for this link, as both endpoints should minimize their distance to H.

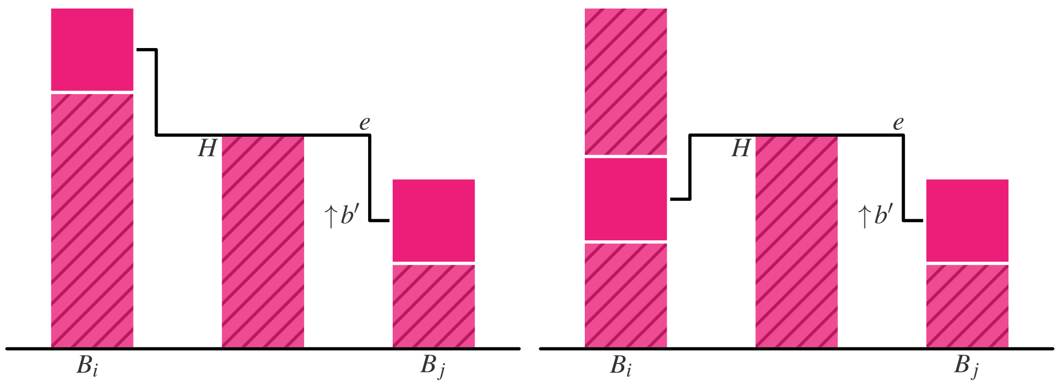


Figure 2 The link is independent, because H is larger than the highest possible y for block b'.

If the previous two cases do not hold, then there is no intermediate bar or both centers may exceed the highest intermediate bar. However, if $\downarrow b \geq \uparrow b'$ (and analogously if $\downarrow b' \geq \uparrow b$), then the orientation of the link is always the same. Hence, b should be placed as low as possible and b' as high as possible. The vertical length is $y(b,k) - y(b',k') = (y(b,k) - t) + (t - y(b',k'))$ for any specific placement of the two blocks and we can minimize their length to an arbitrary intermediate target $t \in [\uparrow b', \downarrow b]$. Hence, the link is independent.

If none of the previous cases hold, the link is *dependent*: its vertical length depends on the relative position of the two blocks, and we cannot generally assign a static target for each block separately. Dependent links come in two variants: *adjacent dependent links* and *non-adjacent dependent links*, depending on whether the two linked bars are consecutive in $\mathcal{B}$.

We abbreviate the various link types as IL (independent link), DL (dependent link), ADL (adjacent dependent link) and NADL (non-adjacent dependent link). We refer to edges also via these types, e.g., a dependent edge is an edge with a dependent link.

▶ **Observation 1.** *There are at most $O(n)$ dependent edges; the dependent edges form an outerplanar subgraph of (V,E).*

Proof sketch. Two bars linked by a dependent link must both be higher than any of its intermediate bars: dependent links and therefore its corresponding edges cannot cross. ◀

▶ **Observation 2.** *The containment on the dependent edges defines a hierarchy.*

Proof sketch. As dependent edges cannot cross, there is at most one minimal dependent edge (under containment) whose span contains that of another dependent edge. ◀

▶ **Lemma 3.** *Given (V,E,w) as an adjacency list with only V in order, we can compute the L- and R-sets for all bars in $O(n+m)$ time, all link types in $O(n^2)$ time and the containment hierarchy of dependent edges in $O(n+m)$ time.*

Proof sketch. To compute the L- and R-sets: the bar order matches the order of L (reversed) and R. Traversing the bars from left to right, allows filling the L and R of the other blocks of the incident links directly.

To compute the link types: use prefix sums of h for the L- and R-sets per bar and a priority queue, to traverse the R-set for each bar, while maintaining the maximum height of intermediate bars by traversing the bars.

To compute the hierarchy: sweep over the bars from left to right, at each bar processing the L- and R-sets in order, while maintaining a stack of the lowest dependent edge. ◀

3 Only independent links

We can easily solve this case via dynamic programming. Observe that the solution is built *vertically* from top to bottom, for reasons that will become clear in the next section.

▶ **Lemma 4.** *Given (V,E,w) with only independent edges, we can minimize the vertical link length in $O(nm)$ time.*

Proof sketch. Each bar can be stacked in isolation, to minimize the total distance of each block to its target. Let $D(p,q)$ denote the minimum cost of stacking blocks $\{l_p, \ldots, l_{|L|}\} \subseteq L$ and blocks $\{r_q, \ldots, r_{|R|}\} \subseteq R$. Since either l_p or r_q must be the bottom block in this subproblem, we compute, in appropriate order, each value in D in $O(1)$ time. ◀

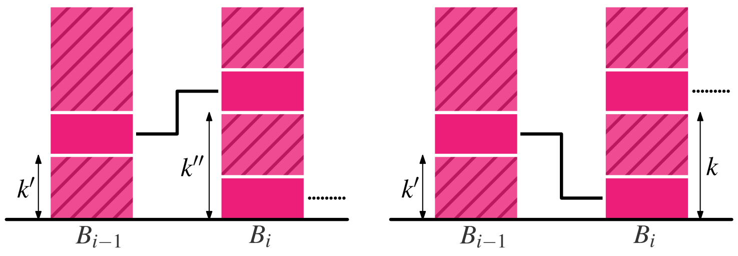


Figure 3 Schematic for computing $D^*(i,k)$ with $k=0$, positioning two blocks (left) and $k>0$, positioning a single block (right) to determine the ADL to the previous bar.

4 Adjacent dependent links

This more general case can be solved efficiently, using the previous result. It uses a *horizontal* dynamic program, which goes through the bars in a shortest-path-like fashion.

In this section, we use the degree d_i of bar B_i to refer to its total number of incident edges and the number of stacked blocks.

▶ **Theorem 5.** *Given (V,E,w) with only independent and adjacent dependent links, we can minimize the vertical link length in $O(nm)$ time.*

Proof sketch. This horizontal recursion $D^*(i,k)$ represents the minimum cost for stacking the blocks in the first i bars, where the ADL in R_i has k blocks from L_i below it; see Fig. 3. As there are only $O(d_{i-1})$ positions to try for an ADL if $k>0$, we can achieve the desired bound by precomputing the cost of independent links with Lemma 4. ◀

▶ **Observation 6.** *If the bars are sorted by height or bar heights have a single local maximum, then all edges have either independent or adjacent dependent links.*

Proof. An NADL requires that all intermediate bars are strictly lower than its endpoints. ◀

5 One-sided non-adjacent dependent links

We call a bar B *one-sided* if L or R does not contain an NADL. When all bars are one-sided, no two spans of NADLs share only an endpoint. By Observation 2, the NADLs induce a hierarchy as well. We call an NADL *constrained* if it shares a bar with its parent NADL, and *free* otherwise; see also Fig. 4. We partition the general hierarchy such that each free NADL forms a root of a (binary) tree, with only its descendant constrained NADLs.

Consider some NADL $\bar{e}$ that connects bars B_i and B_j and fix the position of its blocks in some position. The *associated blocks* of such a position are all blocks in B_i and B_j that are lower than the endpoints of the positioned parent NADL of $\bar{e}$, and all blocks that are in bars at the endpoints of a descendant constrained NADL.

For a given position of an NADL $\bar{e}$, the cost of its associated blocks is independent of the cost of any other blocks as there are no ADLs. So, we solve the interior of an NADL, recursively. Via dynamic programming, we then obtain the following result.

▶ **Theorem 7.** *Given (V,E,w) with one-sided bars and only independent and non-adjacent dependent links, we can minimize the vertical link length in $O(n^4)$ time.*

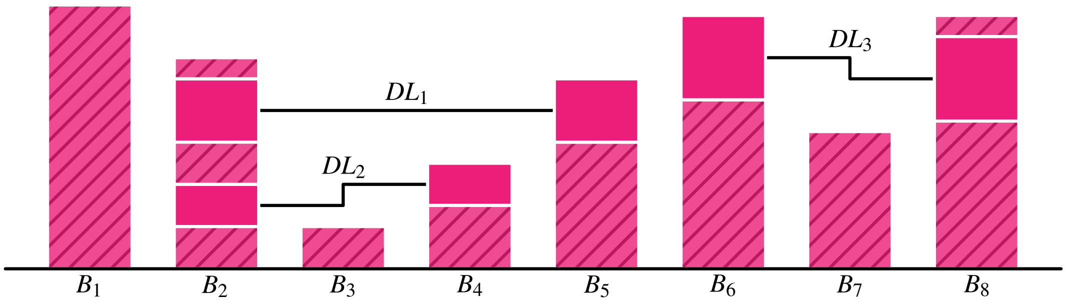


Figure 4 A linked bar chart where all bars are one-sided; only the NADLs are drawn. Links DL_1 and DL_3 are free; DL_2 is constrained by its parent DL_1.

Proof sketch. Bars without NADLs can be solved via Lemma 4. Free NADLs can be positioned in any which way as to optimize the cost of its associated blocks, while a constrained NADL needs to stay below the position of its parent. We use dynamic programming to compute the optimal costs of NADLs, using the recursion to keep track of the constraint that may be given by a parent NADL.

Specifically, we use a dynamic program $D^{**}(\bar{e}, k)$ to represent the minimal total cost for the blocks associated with $\bar{e}$, given that it may use at most k blocks of the other set below it (the constraint of the parent). See Fig. 5 for a sketch of the recursion. Using techniques based on the dynamic program for Lemma 4, we can compute each such case in $O(n^2)$ time. With $O(n)$ links (Observation 1) and at most n values for k, the claimed time bound follows. ◀

6 Conclusion

A natural next step is to combine the techniques of Theorems 5 and 7 for the unified case, which models e.g. situations with a single local minimum in bar height. Relevant future work further includes the unrestricted case with a fixed bar order, optimizing the bar order and considering other quality measures than (only) vertical link length.

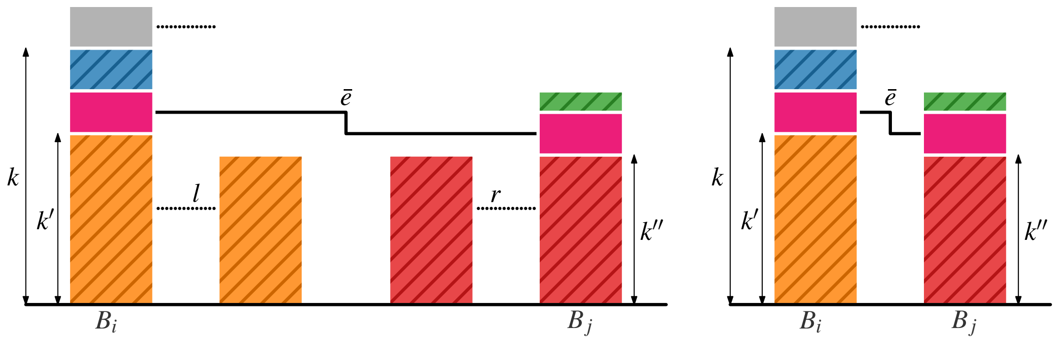


Figure 5 Schematic illustrating how to solve $D^{**}(\bar{e}, k)$ for the case of two descendants (left) and no descendants (right). Gray: constraint k imposed by a parent NADL. Pink: the cost for positioning $\bar{e}$. Orange and red: costs for associated blocks below the endpoints of $\bar{e}$. Blue and green: costs for independent blocks above the endpoints of $\bar{e}$.

References

1 Greg N. Frederickson and Susanne E. Hambrusch. Planar linear arrangements of outerplanar graphs. *IEEE Transactions on Circuits and Systems*, 35(3):323–333, 1988.

2 Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1990.

3 Sumio Masuda, Toshinobu Kashiwabara, Kazuo Nakajima, and Toshio Fujisawa. On the NP-completeness of a computer network layout problem. In *Proc. IEEE International Symposium on Circuits and Systems*, 1987.

4 Sandra L. Mitchell. Linear algorithms to recognize outerplanar and maximal outerplanar graphs. *Information Processing Letters*, 9(5):229–232, 1979.

5 Nathan van Beusekom, Steven Chaplick, Amy Griffin, Martin Krzywinski, Marc van Kreveld, and Hsiang-Yun Wu. Set size visualization with dependent and independent uncertainties. In *Set Visualization and Uncertainty*, number 11 in Dagstuhl Reports, pages 90–94, 2023.

Extending the contour tree distance to embedded graphs

Maike Buchin[1] and Lea Thiel[2]

1 **Ruhr University Bochum**
`maike.buchin@rub.de`

2 **Ruhr University Bochum**
`lea.thiel@rub.de`

Abstract

In order to compare embedded graphs, many different distance measures have already been defined, one of which is the contour tree distance. This measure was originally defined for the comparison of contour trees of two surfaces and later extended naturally for the comparison of embedded trees and graphs. We show that this first definition for graphs does not fulfill the properties of a metric and propose an alternative definition that does. We also establish the relationship of the contour tree distance to the Fréchet distance for curves and the graph distance for embedded graphs.

1 Introduction

For embedded graphs, there are many measures to determine their similarity. Buchin et al. [4] survey several of these measures and which properties of a metric they fulfill. One of the measures is the contour tree distance. It was originally introduced by Buchin, Ophelders and Speckmann [3] and was motivated by the computation of the Fréchet distance of surfaces. It was introduced to compare the contour trees of two surfaces. Its goal is to assign parts of the contour trees that are similar to each other, but also take into account the structure of the trees by only matching connected parts. As such it is a natural measure also for comparing embedded trees in general. Since each point in a tree is assigned to a connected subtree of the other graph and this assignment must be connected overall, it is implicitly required that connected parts of one tree are assigned to connected parts of the other tree.

Later, this distance measure was naturally extended by [4] to embedded trees and graphs. This provides a symmetric distance measure on trees that takes the structure of the trees into account. However, this extension of the definition does not map the structure of general graphs in the same way as for trees: Cycles in the graph can lead to connected parts of one graph being mapped to non-connected components of the other graph.

In this paper, we consider undirected, connected graphs embedded in $\mathbb{R}^2$. In the following, we show that the contour tree distance for general graphs as defined in [4] does not fulfill the triangle inequality. We propose an alternative definition for the contour tree distance on embedded graphs, which leads to the same distance values on trees as the original definition, but also fulfills the properties of a metric for embedded graphs. In addition, we establish a relationship between the contour tree distance and the Fréchet distance on curves [2] and the graph distance [1], which is also based on the Fréchet distance.

2 Contour Tree Distance

Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be two undirected, connected graphs with vertices embedded as points in $\mathbb{R}^2$ that are connected by crossing free straight-line edges. In this

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

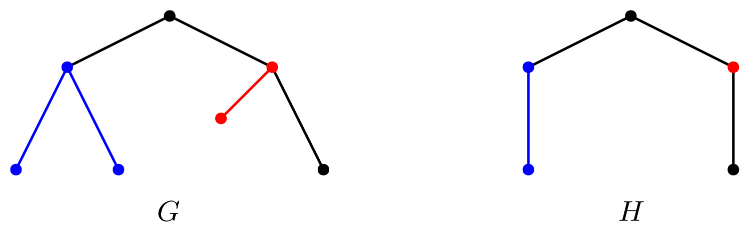


Figure 1 A matching on two trees G and H. The black parts of the trees are matched point by point with the next point in the other tree. There is no similar part in H to the red edge in G, which is why it is "contracted" to the red node in H. The blue parts of the graphs are in a similar position in the trees, but their structure is different. Therefore, each point in the blue part of G is matched with the entire blue part in H and vice versa.

section, we introduce the definition of the contour tree distance on embedded graphs as proposed by [4] and show that it does not fulfill the triangle inequality on graphs. For this, we first define a class of matchings τ, which specifies the assignment between the graphs.

▶ **Definition 2.1.** A matching $\tau \in M_t(G, H)$ between graphs G and H has the following properties:

1. τ is a connected subset of $G \times H$.
2. For each $x \in G$, $\tau \cap (\{x\} \times H)$ (projected onto H) is a non-empty, connected subset of H.
3. For each $y \in H$, $\tau \cap (G \times \{y\})$ (projected onto G) is a non-empty, connected subset of G.

Note that this definition is a natural generalization of the original definition for contour trees [3]. The contour tree distance is then defined as the largest distance between two matched points in an optimal matching.

▶ **Definition 2.2.** The *contour tree distance* d_C between graphs G and H is defined as

$$\delta_C(G, H) = \inf_{\tau \in M_t(G,H)} \sup_{(x,y)\in\tau} \|x - y\|_2,$$

where $\|x - y\|_2$ is the Euclidean distance between the embeddings of x and y.

The measure thus assigns similar parts of the trees to each other while also incorporating the structure of the entire trees, see for example Figure 1.

However, as can be seen in Figure 2 the structure of cycles is not preserved by this matching definition. The preservation of the tree structure was realized by the connectedness of the matching, but we can interrupt cycles at one point without destroying the entire connection. This leads in particular to the fact that this definition does not fulfill the triangle inequality on graphs, which can be seen in the example from Figure 3. We propose the following alternative definition, which also takes into account the structure for cycles and is thus closer to the intuition of the original distance for contour trees:

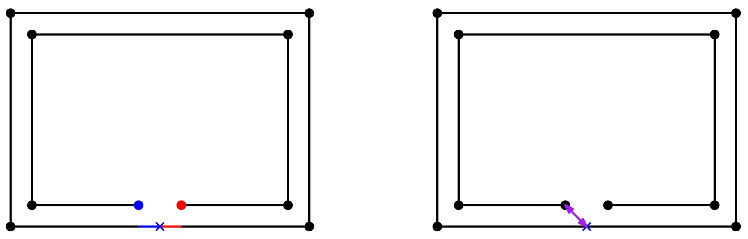


Figure 2 The structure of cycles is not preserved with the definition for the contour tree distance on graphs proposed in [4]. The left-hand image shows a minimal matching: The blue end point of the path is matched with the blue marked part of the edge of the second graph, including the point exactly in the middle of the two end points of the path (blue cross). The right end point is matched with the part of the graph marked in red. The right-hand image shows the maximum distance between two matched points. The overall distance is small, even though the structure of the graphs is different.

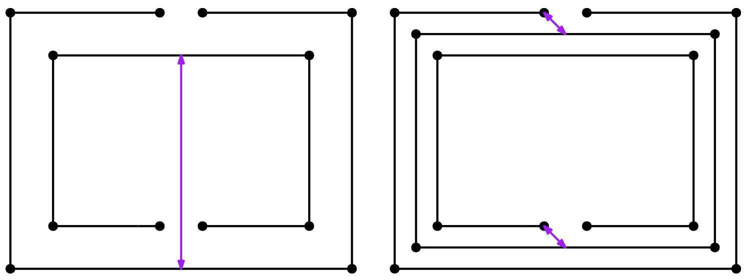


Figure 3 An example where the triangle inequality is not fulfilled for the original extension of the definition to graphs. The two paths in the left-hand image have a large contour tree distance, which here equals the Fréchet distance of the paths. However, the distance of both paths to a cycle placed in between is small, as in the right-hand image, which violates the triangle inequality.

▶ **Definition 2.3.** A matching $\tau \in M_g(G, H)$ between graphs G and H has the following properties:

1. τ is a connected subset of $G \times H$.
2. For each connected subset $X \subseteq G$, $\tau \cap (X \times H)$ projected onto H is a non-empty, connected subset of H.
3. For each connected subset $Y \subseteq H$, $\tau \cap (G \times Y)$ projected onto G is a non-empty, connected subset of G.

This new definition of the matching class M_g corresponds to the definition of matchings M_t on trees.

▶ **Lemma 2.4.** *Let G and H be trees. Then $M_g(G, H) = M_t(G, H)$.*

Proof. Let $\tau' \in M_g(G, H)$, where G and H are trees. Since conditions 2 and 3 are fulfilled in particular for individual points of the two graphs, both conditions also apply to matchings from M_g and $\tau' \in M_t(G, H)$ follows.
Now let $\tau'' \in M_t(G, H)$. For a contradiction, we assume that a connected subset X exists that violates Condition 2 of $M_g(G, H)$. Then either an $x \in X$ exists, which is not assigned to any part of H, which leads to a contradiction due to Condition 2 of $M_t(G, H)$, or X is mapped to a non-connected subset of H. Let y_1 and y_2 be two points from different

components of $\tau'' \cap (X \times H)$ projected onto H. Then x_1 and x_2 exist with $(x_1, y_1) \in \tau''$ and $(x_2, y_2) \in \tau''$. Since τ'' is connected according to Condition 1 of $M_t(G, H)$, a path P must exist between (x_1, y_1) and (x_2, y_2) in τ''. We now want to shorten this path until its projection onto G corresponds to the unique simple path between x_1 and x_2. To do this, we proceed as follows: Let $\Pi_G(P)$ be the projection of P onto G. Let x be a point on G that is visited twice by $\Pi_G(P)$. Then there is a partial path $\tilde{P}$ of P that starts at a point (x, y) and ends at (x, y'). We can now shorten this partial path by staying at the point x on G and using the unique simple path from y to y' on H. This path is also in our matching τ'', since according to Condition 2 each point of G is matched with a connected subtree of H. In this way, we can now replace each detour of our path on G and obtain a path in our matching τ'' that corresponds to the simple path between x_1 and x_2 projected onto G. Since X is connected and G is a tree, the unique path from x_1 to x_2 is in X and therefore P should be in $\tau'' \cap (X \times H)$. However, since this is not connected, we get a contradiction. Thus, condition 2 of $M_g(G, H)$ applies and, analogously, it can be shown that condition 3 is fulfilled. ◀

With the new class of matchings on graphs as in Definition 2.3, the proof from [4] can now be used to show that the contour tree distance is a metric on embedded graphs.

3 Relationship to the Fréchet distance and the graph distance

In this section, we relate the contour tree distance to two known measures and show that it is in a way a generalization of the Fréchet distance to graphs. We start by showing:

▶ **Theorem 3.1.** *The contour tree distance on paths P and Q, when the start and end points are matched, corresponds to the Fréchet distance of the two paths.*

Proof. Assume P and Q have contour tree distance δ_C, i.e. there is a matching $\tau \in M_t(P, Q)$ where matched points have at most distance δ_C and the start and end points of the paths are matched. We want to show that traversals of P and Q with at most distance δ_C exist. Since the start points s_1 of P and s_2 of Q as well as the end points t_1 of P and t_2 of Q are matched, and τ is connected, there is a path in τ from (s_1, s_2) to (t_1, t_2). If we now project this path onto P and Q respectively, we obtain common traversals of the two curves. We can eliminate possible backward movements in a similar way as in the previous lemma. If a point x on a curve is traversed twice, we change the traversals and wait at this point x while we traverse the part between the two visits at x on the other curve. This part must be completely matched with the point x, since each point is matched with a connected subset of the other path. The distance is at most δ_C at any time, as we only traverse matched points simultaneously. The Fréchet distance is therefore at most δ_C.

Assuming P and Q have Fréchet distance δ, then there are traversals of the two curves with distance at most δ. We are looking for a matching $\tau \in M_t(P, Q)$ of the two curves with width δ such that τ, $\tau \cap (\{x\} \times Q)$ projected on Q and $\tau \cap (P \times \{y\})$ projected on P are connected and non-empty for all x and y. To do this, we match exactly the points of the two curves that are traversed at the same time. We now check the conditions of a matching:

1. Since the basis for the matching is a traversal, τ is connected.
2. $\tau \cap (\{x\} \times Q)$ is not empty, as every point of the curve is traversed during a traverse. As one is not allowed to move backwards for the Fréchet distance when traversing, $\tau \cap (\{x\} \times Q)$ projected on Q is connected.
3. Analogue to **2**.

The distance between the matched points is at most δ, as only simultaneously traversed points were matched. τ is therefore a valid matching for the contour tree distance, where matched points have at most distance δ, so the contour tree distance is at most δ. Overall, it follows that the contour tree distance on paths corresponds to the Fréchet distance. If one does not require the start and end points of the respective paths to be matched, the contour tree distance may be smaller. ◀

Since a matching assigns connected subsets of one graph to connected subsets of the other graph, the following lemma is easily obtained:

▶ **Lemma 3.2.** *Consider two graphs G and H and a matching $\tau \in M_g(G, H)$ of width δ. Let (x_1, y_1) and (x_2, y_2) from τ be two matched pairs of points. Then for every simple path $P(x_1, x_2) \in G$ there exists a path $P'(y_1, y_2) \in H$ with $\delta_F(P, P') \leq \delta$.*

Finally, we compare the contour tree distance with another well-known measure for graphs, the *graph distance*, which is based on the Fréchet distance and was introduced by [1]. The idea of this distance is to map one of the graphs onto a subgraph of the other that is as similar as possible. For this, a mapping $s\colon G \to H$ is called a *graph mapping* if it maps each vertex $v \in V_G$ to a point $s(v)$ on an edge of H, and it maps each edge $\{u, v\} \in E_G$ to an arbitrary simple path from $s(u)$ to $s(v)$ in the embedding of H.

▶ **Definition 3.3.** The *directed graph distance* $\vec{\delta}_G$ of two graphs G and H is defined as

$$\vec{\delta}_G(G, H) = \inf_{s:G\to H} \max_{e\in E_G} \delta_F(e, s(e)),$$

where s ranges over graph mappings from G to H, δ_F denotes the Fréchet distance, and e and its image $s(e)$ are interpreted as curves in the plane. The *undirected graph distance* is defined as $\delta_G(G, H) = \max(\vec{\delta}_G(G, H), \vec{\delta}_G(H, G))$.

We can now show that the contour tree distance is "stronger" than the graph distance in the sense that the contour tree distance is always at least as large as the graph distance:

▶ **Theorem 3.4.** *If G and H are two embedded graphs, then $\delta_G(G, H) \leq \delta_C(G, H)$.*

Proof. Let $\delta_C(G, H) = \delta$ and $\tau \in M_g(G, H)$ be a matching that realizes the width δ. We now construct a mapping $s\colon G \to H$ of width δ. Each node v of G is matched to a connected part Y of the graph H by τ. We now map v to any point from Y. This means that all nodes are mapped to a unique point, as required for the graph distance. The edges must now be mapped to simple paths: We consider an edge e in G with the boundary nodes u and v. Since the matching τ matches every connected subset of G, in particular our edge e, with a connected part of H, there is a simple path in $\tau \cap (e \times H)$ that starts in $\tau \cap (\{u\} \times H)$ and ends in $\tau \cap (\{v\} \times H)$. Since $\tau \cap (\{u\} \times H)$ and $\tau \cap (\{v\} \times H)$ projected on H are connected, we even find a simple path p from $s(u)$ to $s(v)$ in $\tau \cap (e \times H)$. We now map our edge e onto this path p. It remains to show that this path p has a maximum Fréchet distance of δ to e. This holds because the contour tree distance on paths corresponds to the Fréchet distance if the end points of the paths are matched, which is the case here. Overall, we therefore obtain a graph mapping $s\colon G \to H$ of width δ. Similarly, we can construct a mapping $s'\colon H \to G$ of width δ. Thus, the claim is true. ◀

Note that there are also graphs for which the graph distance is arbitrarily smaller than the contour tree distance, see Figure 4.

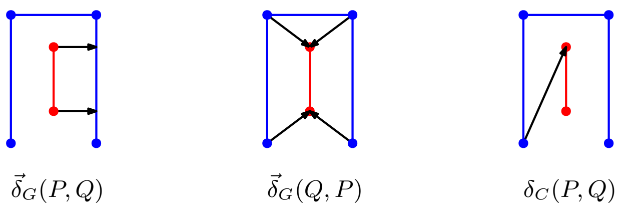


Figure 4 Two paths P (red) and Q (blue), where the graph distance is smaller than the contour tree distance. The black arrows show the respective maximum distance and, on the left and in the center, also the points the nodes are mapped to. If P and the first and last edge of Q are extended equally in the vertical direction, the contour tree distance becomes arbitrarily large, while the directed graph distances remain the same.

4 Conclusion and open questions

We have shown that the contour tree distance with the new extension of the definition fulfills the properties of a metric on embedded graphs. Moreover, we have shown connections of the contour tree distance and the Fréchet distance on curves, as well as that the contour tree distance is "stronger" than the graph distance in the sense that the contour tree distance for two graphs is always at least as large as the graph distance. However, it has been shown [3] that the computation of the original distance for contour trees is already NP-complete, which can be transferred to embedded trees. The computation of the graph distance is also NP-complete for embedded graphs, but can still be computed in polynomial time for trees [1]. It would be interesting to find a distance that lies between the graph distance and the contour tree distance. In other words, a natural symmetrical distance, just like the contour-tree distance, with the strongest possible requirements for maintaining the structure, but which, like the graph distance, can be computed efficiently, at least for trees. Such a distance could possibly be used to efficiently approximate the contour tree distance.

References

1 Hugo A. Akitaya, Maike Buchin, Bernhard Kilgus, Stef Sijben, and Carola Wenk. Distance measures for embedded graphs. *Computational Geometry*, 95:101743, 2021. `doi:10.1016/j.comgeo.2020.101743`.

2 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(1-2):75–91, 1995.

3 Kevin Buchin, Tim Ophelders, and Bettina Speckmann. Computing the Fréchet distance between real-valued surfaces. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, page 2443–2455, USA, 2017.

4 Maike Buchin, Erin Chambers, Pan Fang, Brittany Terese Fasy, Ellen Gasparovic, Elizabeth Munch, and Carola Wenk. Distances between immersed graphs: Metric properties. *La Matematica 2*, (1):197–222, 2023. `doi:10.1007/s44007-022-00037-8`.

Dynamic Maintenance of a Learned Index

Emil Toftegaard Gæde, Ivor van der Hoog, Eva Rotenberg, and Tord Stordalen

Technical University of Denmark, Lyngby, Denmark
{etoga,idjva,erot,tjost}@dtu.dk

Abstract

Consider a dynamic set S of n distinct positive integers and some integer $\varepsilon > 0$. Denote by F_S the two-dimensional point set obtained by mapping each $s \in S$ to $(\texttt{rank}(s), s)$. We dynamically maintain an xy-monotone set of line segments f, such that for all $p \in F_S$, the horizontal line segment with radius ε centred at p intersects a line in f. Our algorithm has $O(\log^2 n)$ worst-case update time and maintains a $\frac{3}{2}$-approximation of the minimum-cardinality set of line segments with this property.

Related Version http://arxiv.org/abs/2503.05007

1 Introduction

Consider a dynamic set S of n distinct positive integers. For any value q, we denote by $\texttt{rank}(q)$ the rank of q if inserted in the sorted order of S. For a fixed integer ε, a *learned index* [4, 8, 9, 11, 12] is a function $h : \mathcal{U} \to [0, n]$ where $\forall q \in \mathcal{U}$, $h(q) \in [\texttt{rank}(q)-\varepsilon, \texttt{rank}(q)+\varepsilon]$. Ferragina and Vinciguerra [8] observe that such a function h is of a geometric nature:

Denote by F_S the two-dimensional point set obtained by mapping each $s \in S$ to $(\texttt{rank}(s), s)$. For a fixed integer ε, we define an *ε-cover* f of S as an xy-monotone set of line segments such that for each point $p \in F_S$, the horizontal line segment of radius ε intersects a line segment in f. Ferragina and Vinciguerra [8] show that an ε-cover implies a learned index, whose complexity is linear in the number of line segments in f (Figure 1). Thus, we are interested in computing an ε-cover of minimum cardinality. Ferragina and Vinciguerra [8] show an algorithm that, given S in its sorted order, computes in linear time an ε-cover of f whose cardinality is a $\frac{3}{2}$-approximation of the minimum-cardinality ε-cover.

In applied settings, recomputing static solutions might be prohibitive. We use classical techniques from computational geometry to dynamically maintain a $\frac{3}{2}$-approximate minimum-cardinality ε-cover in $O(\log^2 n)$ time per insertion in S.

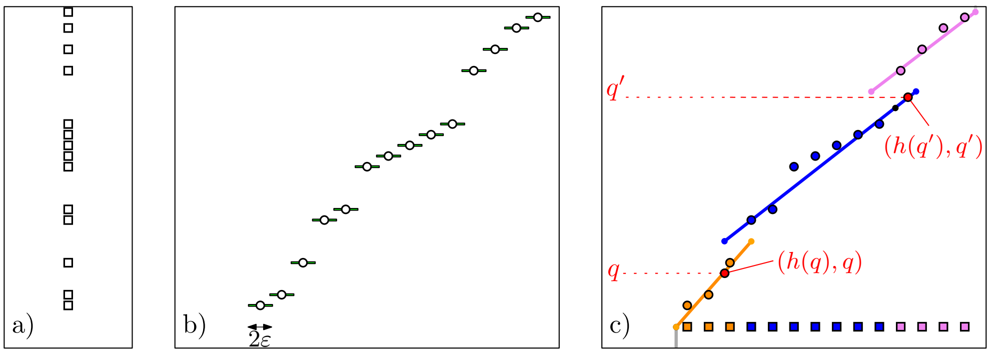


Figure 1 (a) a set of n values S. (b) The xy-monotone point set F_S. (c) an ε-cover f of F_S

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

2 Testing whether a set can be ε-covered by a single segment

To solve our problem, we first consider the following subproblem: given ε, a set of integers S with no duplicates, and the edges of $\mathrm{CH}(F_S)$ in a balanced binary tree, can we compute in $O(\log n)$ time whether an ε-cover f of complexity 1 exists and, if so, can we output f?

Denote by A (or, B) the point set obtained by shifting each $p \in F_S$ rightwards by ε and adding $(\infty, -\infty)$ (or, leftwards by ε, adding $(-\infty, \infty)$)

▶ **Lemma 2.1.** *Let ℓ be a line with positive slope. Then ℓ is an ε-cover of S if and only if ℓ lies below all points in B and above all points in A.*

Proof. Any line with positive slope lies above $(\infty, -\infty)$ and below $(-\infty, \infty)$. Consider a point $p \in F_S$ and the two corresponding points $l \in A$ and $u \in B$. Denote by C_p a horizontal segment centred at p of radius $\varepsilon > 0$. If ℓ lies below l then all points on ℓ left of l lie below C. If ℓ lies above u then all points on ℓ right of u lie above C. If ℓ lies above l and below u then because ℓ has positive slope, it must intersect C. The statement follows. ◀

▶ **Corollary 2.2.** *A line is an ε-cover of S if and only if it separates $CH(A)$ and $CH(B)$.*

Given $\mathrm{CH}(F_S)$, we can extract $\mathrm{CH}(A)$ and $\mathrm{CH}(B)$ in logarithmic time (we implicitly translate all points horizontally by $+\varepsilon$ or $-\varepsilon$). Chazelle and Dobkin [3, Section 4.2] remark that hull intersection testing, in the negative case, 'can be modified' to output a separating line. However, no explicit algorithm is given. All subsequent 2-dimensional intersection testing algorithms [6, 2, 7, 13, 5, 1], also contain no algorithmic description for obtaining a separating line in the negative case. We provide an adaptation of the $O(\log n)$-time convex hull intersection testing algorithm by Chazelle and Dobkin [2], restricted to convex hulls of edges with positive slope that contain $(\infty, -\infty)$ and $(-\infty, \infty)$. We then develop an algorithm for computing a separating line in the negative case.

3 Separating lines of convex hulls

Consider two convex hulls $CH(A)$ and $CH(B)$, where $CH(A)$ contains $(\infty, -\infty)$, $CH(B)$ contains $(-\infty, \infty)$ and all convex hull edges have positive slope. We present an algorithm based on [2] that compares pairs of edges $(\alpha, \beta) \in CH(A) \times CH(B)$, and decides in $O(\log n)$ time whether these two hulls intersect. For completeness, we show in the full version the correctness of this adaption:

▶ **Theorem 3.1.** *Algorithm 1 outputs in $O(\log n)$ time whether $CH(A)$ and $CH(B)$ intersect.*

Algorithm 1 `intersection_test(edge` $\alpha \in CH(A)$, `edge` $\beta \in CH(B)$)

```
 1: if  $\alpha = null$ OR $\beta = null$ then
 2:     return No
 3: end if
 4: $s(\alpha, \beta) = line(\alpha) \cap line(\beta)$
 5: if $s \in \alpha$ and $s \in \beta$ then
 6:     return Yes
 7: end if
 8: if $\alpha$.slope > $\beta$.slope then                  ▷ .slope denotes the slope of an edge
 9:     if $\alpha$.second.$x$ < $s(\alpha, \beta).x$ then      ▷ .first and .second denote endpoints of edges
10:         return intersection_test($\alpha$.right, $\beta$)  ▷ .left and .right denote children in tree
11:     else if $\beta$.second.$x$ < $s(\alpha, \beta).x$ then
12:         return intersection_test($\alpha$, $\beta$.right)
13:     else if $\alpha$.first.$x$ > $\beta$.second.$x$ AND $\alpha$.first.$y$ > $\beta$.second.$y$ then
14:         return intersection_test($\alpha$, $\beta$.right)
15:     else if $\alpha$.second.$x$ < $\beta$.first.$x$ AND $\alpha$.second.$y$ < $\beta$.first.$y$ then
16:         return intersection_test($\alpha$.right, $\beta$)
17:     else
18:         return yes
19:     end if
20: else if $\alpha$.slope < $\beta$.slope then                 ▷ Symmetric to the above case
21:     if  $\alpha$.first.$x$ > $s(\alpha, \beta)$.x then
22:         return intersection_test($\alpha$.left, $\beta$)
23:     else if  $\beta$.first.$x$ > $s(\alpha, \beta)$.x  then
24:         return intersection_test($\alpha$, $\beta$.left)
25:     else if $\alpha$.first.$x$ > $\beta$.second.$x$ AND $\alpha$.first.$y$ > $\beta$.second.$y$ then
26:         return intersection_test($\alpha$.left, $\beta$)
27:     else if $\alpha$.second.$x$ < $\beta$.first.$x$ AND $\alpha$.second.$y$ < $\beta$.first.$y$ then
28:         return intersection_test($\alpha$, $\beta$.left)
29:     else
30:         return yes
31:     end if
32: else                                   ▷ The parallel edge case, $\alpha$.slope = $\beta$.slope
33:     if $line(\beta)$ is above $line(\alpha)$ then
34:         return No
35:     else if $\alpha$.first.$x$ > $\beta$.second.$x$ AND $\alpha$.first.$y$ > $\beta$.second.$y$ then
36:         return intersection_test($\alpha$.left, $\beta$)
37:     else if $\alpha$.second.$x$ < $\beta$.first.$x$ AND $\alpha$.second.$y$ < $\beta$.first.$y$ then
38:         return intersection_test($\alpha$, $\beta$.left)
39:     else
40:         return No
41:     end if
42: end if
```

Our main focus is on what to do whenever Algorithm 1 outputs that $CH(A)$ and $CH(B)$ do not intersect. In this case, we show how to find a line that separates $CH(A)$ and $CH(B)$ in $O(\log n)$ time. Algorithm 1 outputs *no* in two cases. The first case is the special case where there exist two parallel edges $\alpha \in CH(A)$ and $\beta \in CH(B)$ where $line(\beta)$ lies above

$line(\alpha)$. In this case both $line(\alpha)$ and $line(\beta)$ are a separating line.

The second case is that either argument of the function was *null*. Without loss of generality, we assume that β was null. Then there exist two pairs of edges $(\alpha, \beta), (a, b) \in CH(A) \times CH(B)$ where `intersection_test`(α, β) recurses on β.right and `intersection_test`(a, b) recurses on b.left. Moreover, the edges β and b must share a vertex. We define the following:

▶ **Definition 3.2.** For any edge α, we denote by $\overleftarrow{\alpha}$ and $\overrightarrow{\alpha}$ its two supporting halflines. For edges (β, b) that share a vertex with β left of b, we denote by $w(\beta, b) = \overleftarrow{\beta} \cup \overrightarrow{b}$ their *wedge*.

By keeping track of the traversal of Algorithm 1, we obtain $w(\beta, b)$ at no overhead.

▶ **Lemma 3.3.** *Let Algorithm 1 terminate without finding an intersection between $CH(A)$ and $CH(B)$ and denote by $w(\beta, b)$ the corresponding wedge. Then, the halfline $\overleftarrow{\beta}$ cannot intersect $CH(A)$, and the halfline $\overrightarrow{b}$ cannot intersect $CH(A)$.*

Proof. We first prove that the halfline $\overleftarrow{\beta}$ cannot intersect $CH(A)$. Then there exists some $\alpha \in CH(A)$ where `intersection_test`(α, β) recurses on β.right. Thus, $slope(\alpha) > slope(\beta)$. Define $s(\alpha, \beta) = line(\alpha) \cap line(\beta)$. Observe that `intersection_test`$(\alpha,\ \beta)$ recurses on β.right in two cases. The first case is whenever β.second.$x < s(\alpha, \beta).x$. Since $CH(A)$ lies in the plane upper bounded by $line(\alpha)$ this implies that $\overleftarrow{\beta}$ cannot intersect $CH(A)$.

In the second case, the vertex α.first dominates β.second (Figure 2 (a)). Suppose for the sake of contradiction that $\overleftarrow{\beta}$ intersects $CH(A)$ in some point q left of β.second. Since $line(\beta)$ has positive slope, β.second must dominate q. Consider the convex area G bounded by a curve γ that traverses $CH(A)$ backwards until q, after which it becomes a vertical downward halfline. It follows that β.second is contained in $G \subseteq CH(A)$. This implies that $CH(A)$ and $CH(B)$ intersect which is a contradiction. The argument that shows that $\overrightarrow{b}$ cannot intersect $CH(A)$ is symmetric (see Figure 2 (b)) and can be found in the full version. ◀

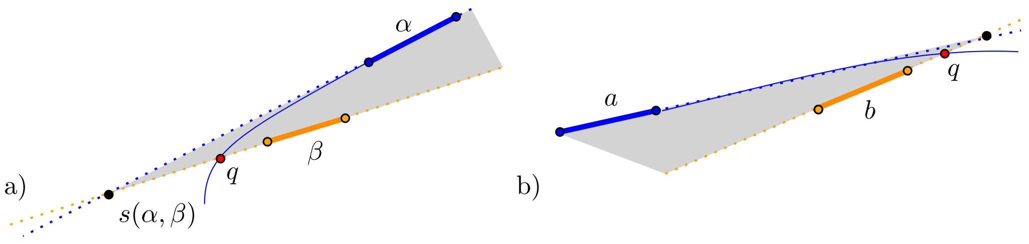


Figure 2 (a) If there exists an edge α of $CH(A)$ that dominates β and $\overrightarrow{\beta}$ intersects $CH(A)$ in a point q then we may argue that β is contained in $CH(A)$. (b) If there exists an edge a of $CH(A)$ that is dominated by an edge b then we make the symmetrical argument.

Given the edge $w(\beta, b)$ we run Algorithm 2, starting with the root of α.

▶ **Lemma 3.4.** *Algorithm 2 outputs an edge in* $\mathrm{CH}(A) \cup CH(B)$ *whose supporting line separates* $\mathrm{CH}(A)$ *and* $\mathrm{CH}(B)$.

Proof. There always exists an edge on $CH(A)$ or $CH(B)$ whose supporting line separates the two convex hulls [3]. Our algorithm always finds either an edge of $CH(A)$, or guarantees that for all edges in $\mathrm{CH}(A)$ their supporting line intersects $w(\beta, b)$. Indeed, since $CH(B)$ is contained in $w(\beta, b)$, a line $line(\alpha)$ separates the two hulls if it does not intersect $w(\beta, b)$.

Whenever $line(\alpha)$ *does* intersect $w(\beta, b)$, either $\overleftarrow{\alpha}$ or $\overrightarrow{\alpha}$ must intersect $w(\beta, b)$. Let $\overleftarrow{\alpha}$ intersect $w(\beta, b)$. Any edge $a \in CH(A)$ succeeding α must have lower slope and so $\overleftarrow{a}$ must

Algorithm 2 `separation_find(wedge` $w(\beta, b)$, `edge` $a \in CH(A)$

```
1: if $\alpha = null$ then
2:     return $line(\beta)$ or $line(b)$
3: end if
4: if $line(\alpha) \cap w(\beta, b) = \emptyset$ then
5:     return $line(\alpha)$
6: else if $\overleftarrow{\alpha} \cap w(\beta, b) \neq \emptyset$ then
7:     return separation_find( $w(\beta, b)$, $\alpha$.left)
8: else
9:     return separation_find($\beta$, $b$, $\alpha$.right)
10: end if
```

intersect $w(\beta, b)$. Similarly, if $\overrightarrow{a}$ intersects $w(\beta, b)$ for any edge $a \in CH(A)$ preceding α, $\overrightarrow{a}$ intersects $w(\beta, b)$. Since A starts with a vertical downwards halfline and ends with a horizontal rightwards halfline, it cannot be that for all edges $a \in CH(A)$ the halfline $\overleftarrow{a}$ intersects $w(\beta, b)$ (the same is true for $\overrightarrow{a}$). Thus, if Algorithm 2 does not output an edge $\alpha \in CH(A)$ then there must exist two consecutive edges (γ, g) on $CH(A)$ such that: for all edges γ' of $CH(A)$ preceding and including γ, $\overrightarrow{\gamma'}$ intersects $w(\beta, b)$, and, for all edges g' of $CH(A)$ succeeding an including g, $\overleftarrow{g}$ intersects $w(\beta, b)$. We conclude the argument:

- If $\overrightarrow{\beta}$ does not intersect $CH(A)$ then by Lemma 3.3, $line(\beta)$ separates $CH(A)$ and $CH(B)$.
- If $\overleftarrow{b}$ does not intersect $CH(A)$ then by Lemma 3.3, $line(b)$ separates $CH(A)$ and $CH(B)$.
- If $\overrightarrow{\beta}$ intersects $CH(A)$ in an edge γ' that equals or precedes γ then the halfline $\overrightarrow{\gamma'}$ intersects $w(\beta, b)$. The halfline $\overrightarrow{\gamma'}$ cannot intersect $\overleftarrow{\beta}$ since γ' is already intersected by $\overrightarrow{\beta}$. So, $\overrightarrow{b}$ intersects $\overrightarrow{\gamma'}$ (Figure 3). In particular, $\overleftarrow{b}$ does not intersect $line(\gamma')$.
  We now note that all edges of $CH(A)$ are contained in the halfplane bounded from above by γ'. And so, $\overleftarrow{b}$ cannot intersect any edge of $CH(A)$. Lemma 3.3 guarantees that $\overrightarrow{b}$ cannot intersect $CH(A)$ and so $line(b)$ separates $CH(A)$ and $CH(B)$.
- If the edge $\overleftarrow{b}$ intersects $CH(A)$ in an edge g' that equals or succeeds g then it follows by symmetry that $line(\beta)$ separates $CH(A)$ and $CH(B)$.
- It cannot be that $\overrightarrow{\beta}$ intersects $CH(A)$ on an edge strictly succeeding γ *and* that $\overleftarrow{b}$ intersects $CH(A)$ in an edge strictly preceding g.

We showed that Algorithm 2 either outputs a edge α where $line(\alpha)$ separates $CH(A)$ and $CH(B)$, or, that either $line(\beta)$ or $line(b)$ separates $CH(A)$ or $CH(B)$. ◀

4 Dynamically maintaining an ε-cover

For an ε-cover f of S, we define $\Lambda(f)$ as the set of pairwise interior-disjoint one-dimensional intervals that correspond to the y-coordinates of segments in f. We denote for integers $a, b \in S$ with $a \leq b$ by $S[a, b]$ the subsequence of S from a to b. For each $[a, b] \in \Lambda(f)$, we maintain a *rank-based convex hull* $T(S[a, b])$ of $S[a, b]$ as described in [10].

▶ **Theorem 4.1.** *We can dynamically maintain an ε-cover f of S in $O(\log^2 n)$ worst-case time. We guarantee that there exists no ε-cover f' of S where $|f| > \frac{3}{2}|f'|$.*

Proof. The proof is illustrated by Figure 4. For any $s, t \in \mathbb{Z}$ with $s \leq t$, we say that $S[s, t]$ is *blocked* if there exists no ε-cover of $S[s, t]$ of size 1. We maintain an ε-cover f where for all

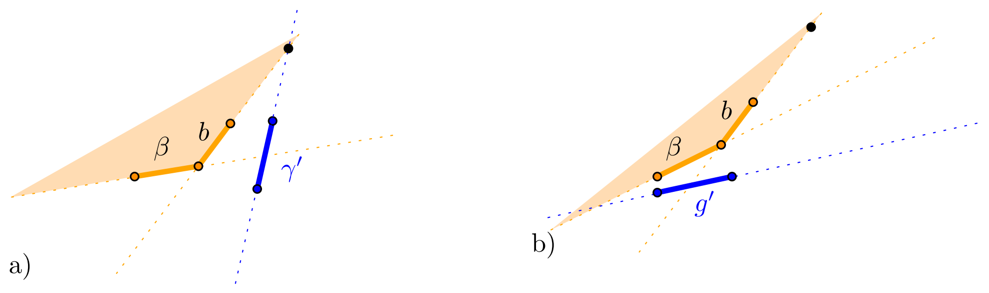


Figure 3 (a) Let $\gamma' \in CH(A)$ be an edge where $\overrightarrow{\gamma'}$ intersects $\overrightarrow{b}$. Then $\overleftarrow{b}$ does not intersect the supporting line of γ'. However, then $\overleftarrow{b}$ cannot intersect any edge of $CH(A)$. (b) Let $g' \in CH(B)$ be an edge where $\overleftarrow{g'}$ intersects $\overleftarrow{\beta}$. Then $\overrightarrow{\beta}$ does not intersect the supporting line of g'.

consecutive $[a, b], [c, d] \in \Lambda(f)$, $S[a, d]$ is blocked. Thereby, $|f| \leq \frac{3}{2}|f'|$ for any ε-cover f' of S (for completeness, we give a proof of this fact in the full version).

We consider inserting a value s into S; deletions are handled analogously. We test in $O(\log n)$ time whether $s \in S$. If so, we reject the update. Otherwise, we search in $O(\log n)$ time for the interval $[a, b] \in \Lambda(f)$ that contains s. If no such interval exists, set $[a, b] = [s, s]$.

We remove $[a, b]$ from $\Lambda(f)$ and insert the intervals $([a, s], [s, s], [s, b])$. We obtain $T(S[a, s])$, $T(S[s, s])$ and $T(S[s, b])$ through the split operation.

Let $([w, x], [y, z], [a, s], [s, s], [s, b], [c, d], [e, f])$ be consecutive intervals in $\Lambda(f)$ and denote $I = ([y, z], [a, s], [s, s], [s, b], [c, d])$ (see Figure 4 (c)). For each $(s, t) \in I$, we have access to $T(S[s, t])$. For any consecutive pair $([s, t], [q, r])$ in I, we may join the trees $T(S[s, t])$ and $T(S[q, r])$ in $O(\log^2 n)$ time to obtain $T(S[s, r])$. We then apply Theorem 3.1 to test in $O(\log^2 n)$ total time whether $S[s, r]$ is *blocked*. If it is not, we replace $[s, t]$ and $[q, r]$ by $[s, r]$. Otherwise, we have found $T(S[s, r])$ and a segment that is an ε-cover of $S[s, r]$.

By recursively merging pairs in I, we obtain in $O(\log^2 n)$ time a sequence I' of intervals $([y, \beta], \ldots, [\gamma, d])$ where consecutive intervals are blocked. Since $[y, z] \subseteq [y, \beta]$, $([w, x], [y, \beta])$ is blocked. Similarly, $([\gamma, d], [e, f])$ must be blocked. We remove the line segments corresponding to I from f and replace them with line segments derived from I' in constant time. As a result, we maintain our ε-cover f and our data structure in $O(\log^2 n)$ total time. ◀

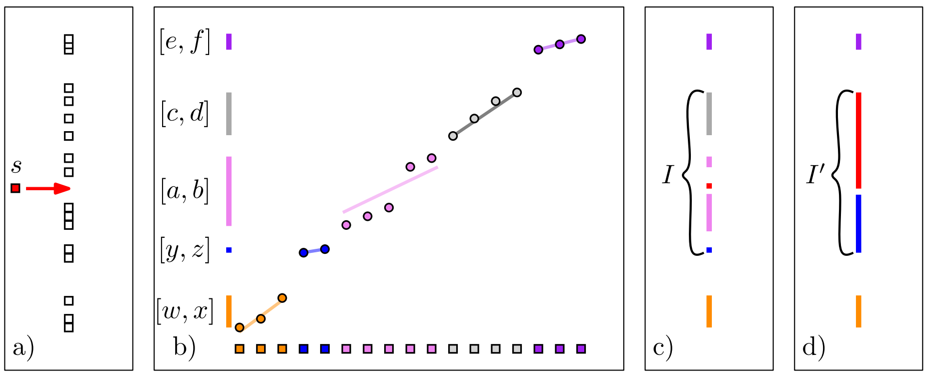


Figure 4 (a) Let S be a set of values and let us insert s. (b) We consider our ε-cover f and five consecutive intervals in $\Lambda(f)$. (c) We create seven intervals by splitting $[a, b]$ on s. (d) By recursively merging intervals in I, we obtain a set of intervals I' where consecutive intervals are blocked.

References

1 Luis Barba and Stefan Langerman. Optimal detection of intersections between convex polyhedra. *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2015.

2 B. Chazelle and D. P. Dobkin. Intersection of convex objects in two and three dimensions. *Journal of the ACM*, 1987. doi:10.1145/7531.24036.

3 Bernard Chazelle and David P Dobkin. Detection is easier than computation. In *ACM Symposium on Theory Of Computing (STOC)*, 1980.

4 Jialin Ding, Umar Farooq Minhas, Jia Yu, Chi Wang, Jaeyoung Do, Yinan Li, Hantian Zhang, Badrish Chandramouli, Johannes Gehrke, Donald Kossmann, et al. Alex: an updatable adaptive learned index. In *ACM International Conference on Management of Data (SIGMOD)*, 2020.

5 David Dobkin and Diane Souvaine. Detecting the intersection of convex objects in the plane. *Computer Aided Geometric Design*, 1991. doi:10.1016/0167-8396(91)90001-R.

6 David P. Dobkin and David G. Kirkpatrick. Fast detection of polyhedral intersection. *Theoretical Computer Science (TSC)*, 1983. doi:10.1016/0304-3975(82)90120-7.

7 David P. Dobkin and David G. Kirkpatrick. Determining the separation of preprocessed polyhedra - a unified approach. In *International Colloquium on Automata, Languages and Programming (ICALP)*, 1990.

8 Paolo Ferragina and Giorgio Vinciguerra. The pgm-index: a fully-dynamic compressed learned index with provable worst-case bounds. *International Conference on Very Large Databases (VLDB)*, 2020.

9 Alex Galakatos, Michael Markovitch, Carsten Binnig, Rodrigo Fonseca, and Tim Kraska. Fiting-tree: A data-aware index structure. In *ACM International Conference on Management of Data (SIGMOD)*, pages 1189–1206, 2019.

10 Emil Gæde, Inge Li Gørtz, Ivor Van Der Hoog, Christoffer Krogh, and Eva Rotenberg. Simple and robust dynamic two-dimensional convex hull. *ACM Symposium on Algorithm Engineering and Experiments (ALENEX)*, 2024.

11 Andreas Kipf, Ryan Marcus, Alexander van Renen, Mihail Stoian, Alfons Kemper, Tim Kraska, and Thomas Neumann. Radixspline: a single-pass learned index. In *International workshop on exploiting artificial intelligence techniques for data management*, 2020.

12 Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *ACM International Conference on Management of Data (SIGMOD)*, 2018.

13 Joseph O'Rourke. *Computational geometry in C (second edition)*. Cambridge University Press, USA, 1998.

On the Shape Containment Problem within the Amoebot Model with Reconfigurable Circuits[*]

Matthias Artmann[1], Andreas Padalkin[2], and Christian Scheideler[3]

1 Paderborn University, Germany
matthias.artmann@upb.de
2 Paderborn University, Germany
andreas.padalkin@upb.de
3 Paderborn University, Germany
scheideler@upb.de

Abstract

In *programmable matter*, we consider a large number of tiny, primitive computational entities called *particles* that run distributed algorithms to control global properties of the particle structure. In this paper, we study the *shape containment problem*, where the particles are given a description of a shape S and have to find maximally scaled representations of S within their initial configuration. We use the *geometric amoebot model* for programmable matter with its *reconfigurable circuit extension*, which allows the instantaneous transmission of primitive signals on connected subsets of particles. We first prove a lower runtime bound of $\Omega(\sqrt{n})$ synchronous rounds for the general problem, where n is the number of particles. Then, we construct the class of *snowflake* shapes and its subclass of *star convex* shapes, and present solutions for both. Let k be the maximum scale of the considered shape in a given amoebot structure. If the shape is star convex, we solve it within $\mathcal{O}(\log^2 k)$ rounds. If it is a snowflake but not star convex, we solve it within $\mathcal{O}(\sqrt{n}\log n)$ rounds.

Related Version A full version of the paper can be found on arXiv: https://arxiv.org/abs/2501.16892.

1 Introduction

Programmable matter envisions a material that can change its physical properties in a programmable fashion [15] and react to external stimuli. It is typically viewed as a system of many identical micro-scale computational entities called *particles*. Potential applications include minimally invasive surgery, maintenance, exploration, and manufacturing. While significant progress is being made in the field of micro-scale robotics [16, 3], the fundamental capabilities and limitations of such systems are studied in theory using various models [14].

In the *amoebot model* of programmable matter, the particles are called *amoebots* and are placed on the nodes of a graph. We assume that the occupied nodes form a connected subgraph. Since information can only travel through the edges of the graph, there is a natural lower bound of $\Omega(D)$ for many problems, where D is the diameter of this subgraph.

Motivated by this, we consider the *reconfigurable circuit extension* of the model, which allows better results with reasonable modifications. In this extension, the amoebots can construct simple communication networks called *circuits* on connected subgraphs of the structure and broadcast primitive signals on the circuits instantaneously. This allows polylogarithmic solutions to many problems, e. g., leader election, consensus, shape recognition, and shortest path forest construction [8, 12, 11].

[*] This work was supported by the DFG Project SCHE 1592/10-1. We want to thank Daniel Warner for his guidance and helpful discussions.

The *shape formation problem*, where the amoebot structure must reconfigure itself into a given shape, is a standard problem of particular interest [14]. We study the related *shape containment problem*: Given a shape S, the amoebots must find the maximum scale at which S can be placed within their structure and identify all valid placements at this scale. This can be useful for shape formation by *self-disassembly*, i. e., disconnecting all amoebots that are not part of a selected placement of the shape from the structure [10, 9]. The problem can also be interpreted as a discrete variant of the *polygon containment problem* in classical computational geometry, which has been studied extensively [4, 13]. To our knowledge, there are no distributed solutions that apply to the amoebot model.

2 Geometric Amoebot Model

The *geometric amoebot model*, as introduced in [6], places $n \in \mathbb{N}$ amoebots on the infinite regular triangular grid graph $G_\Delta = (V_\Delta, E_\Delta)$. Each amoebot occupies one node and each node is occupied by at most one amoebot. We call the set of occupied nodes $A \subset V_\Delta$ the *amoebot structure* and assume that its induced subgraph is connected (see Fig. 1a). The amoebots are identical and anonymous finite state machines. Although the model allows amoebots to perform movements, we only consider static amoebot structures in this paper.

3 Reconfigurable Circuit Extension

The *reconfigurable circuit extension* [8] places c *external links* on every edge connecting two adjacent amoebots $u, v \in A$. The end points of an external link are called *pins*. For each link, one pin is owned by u and one is owned by v. The constant c is an algorithm design parameter and is the same for all amoebots. In this paper, we use $c = 2$, which is the least number of pins required by the *PASC* algorithm [8], the central primitive for our results.

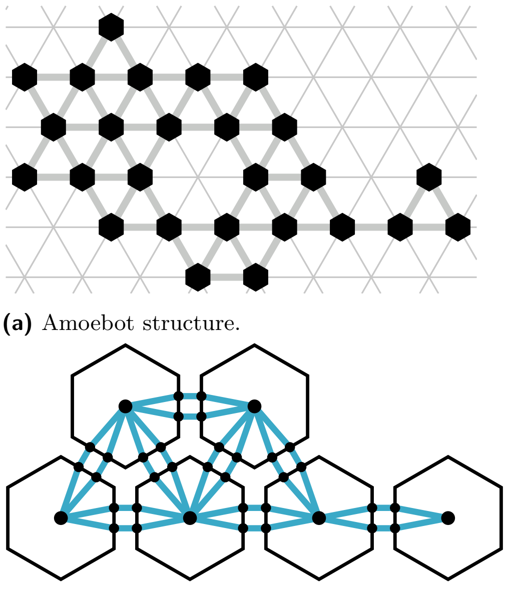
Let $P(u)$ be the set of pins owned by $u \in A$. Each amoebot u partitions $P(u)$ into pairwise disjoint subsets $Q \subseteq P(u)$ called *partition sets* to define its *pin configuration* $\mathcal{Q}(u)$ such that $P(u) = \bigcup_{Q \in \mathcal{Q}(u)} Q$. Let $\mathcal{Q} := \bigcup_{u \in A} \mathcal{Q}(u)$ be the set of all partition sets in the structure. Two partition sets $Q \in \mathcal{Q}(u)$ and $Q' \in \mathcal{Q}(v)$ of neighboring amoebots u and v are *connected* if there is an external link with one pin in Q and one pin in Q'. Let $E_\mathcal{Q}$ be the set of these connections. We call each connected component of the undirected graph $G_\mathcal{Q} := (\mathcal{Q}, E_\mathcal{Q})$ a *circuit* (see Figs. 1c,d). For example, if each amoebot collects all pins in a single partition set, there is only one circuit, spanning the whole structure (a *global circuit*).

During its activation, each amoebot can establish an arbitrary new pin configuration and send primitive signals called *beeps* on any selection of its partition sets. A beep is broadcast to the circuit containing the partition set it was sent on. It is received by all partition sets of that circuit in the next round. An amoebot can tell which of its partition sets have received a beep, but it has no information on the identity, location, or number of beep origins.

The activation model is *fully synchronous*: Time is divided into synchronous *rounds* so that in each round, all amoebots are activated simultaneously. During its activation, an amoebot can update its state, change its pin configuration, and send beeps, all depending on its current state, received beeps from the last round, and observations about its neighbors. We measure the time complexity of an algorithm by the number of rounds it requires.

4 Problem Statement

Consider the embedding of G_Δ into $\mathbb{R}^2$ such that the edges form unit triangles and one node is placed on the origin (see Fig. 1b). We obtain three coordinate axes and six *cardinal*



(a) Amoebot structure.

(b) Grid axes and cardinal directions.

(c) Amoebot structure with one global circuit.

(d) Amoebot structure with six circuits.

Figure 1 (a) An amoebot structure in the triangular grid. Amoebots are black hexagonal nodes and neighbors are connected by thick edges. (b) The axes and cardinal directions in the triangular grid and the unit vector in the East direction. (c, d) The reconfigurable circuit extension for $c = 2$. Amoebots are drawn as hexagons, pins are black circles on their borders and partition sets are drawn as black circles inside the hexagons. The partition sets are connected to the pins they contain. Partition sets in the same circuit have lines of the same color.

directions $\mathcal{D} = \{\mathrm{E}, \mathrm{NE}, \mathrm{NW}, \mathrm{W}, \mathrm{SW}, \mathrm{SE}\}$. A *shape* $S \subset \mathbb{R}^2$ is a finite union of nodes, edges and faces of the embedded grid graph (see Fig. 2, c. f. [7, 8]). Edges contain their endpoints and faces contain their enclosing edges. A shape must be connected but it may contain holes, i. e., $\mathbb{R}^2 \setminus S$ might not be connected.

We define *translation* and *scaling* operations on shapes as follows: For $t \in V_\Delta$ and $k \in \mathbb{N}_0$, let $S + t := \{p + t \mid p \in S\}$ and $k \cdot S := \{k \cdot p \mid p \in S\}$, where t is interpreted as a point in $\mathbb{R}^2$ to simplify notation. For $r \in \mathbb{Z}$, let $S^{(r)}$ denote S after r counter-clockwise rotations by 60° around the origin, defining a *rotation* operation. Observe that $r \in \{0, \ldots, 5\}$ is sufficient.

Let $V(S) \subset V_\Delta$ be the set of nodes contained in S. For simplicity, we assume $(0, 0) \in V(S)$ for any given shape S and call this node the *origin* of S. A *valid placement* of S in an amoebot structure A is an amoebot $p \in A$ with $V(S + p) \subseteq A$. Let $\mathcal{V}(S, A)$ be the set of all valid placements of S in A. The *maximum scale* $k_{\max}(S, A)$ of a shape S in an amoebot structure

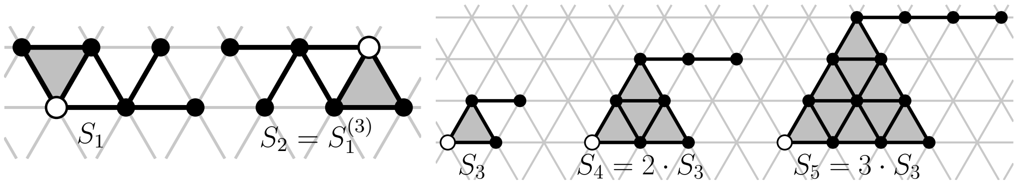


Figure 2 Examples of shapes and shape operations. Each shape is composed of grid nodes, edges and faces. The origin of a shape is highlighted in white. S_2 is the result of rotating S_1 by 180° (and translating it for better visibility). The shapes S_3, S_4 and S_5 illustrate the scaling operation.

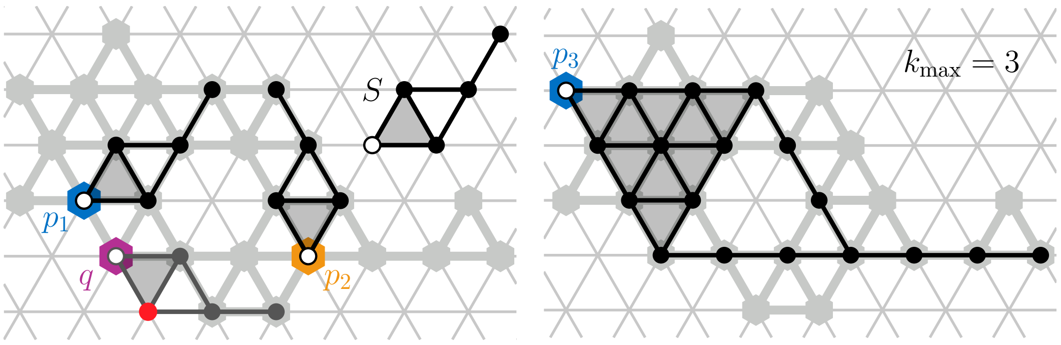


Figure 3 Valid placements of a shape S in the amoebot structure A from Fig. 1a. p_1 and p_2 are valid placements of S and $S^{(1)}$, respectively. q is not a valid placement of $S^{(5)}$ because one node of $V(S^{(5)} + q)$ is not contained in A. p_3 is the only valid placement of $3 \cdot S^{(5)}$ and there are no valid placements for any scale $k > 3$, so we have $k_{\max}(S, A) = 3$. A shape containment algorithm allows p_3 to determine $p_3 \in \mathcal{V}(k_{\max} \cdot S^{(5)}, A)$ and rules out all other amoebots and rotations.

A is the maximum $k \in \mathbb{N}_0$ for which there exists an $r \in \{0, \ldots, 5\}$ with $\mathcal{V}(k \cdot S^{(r)}, A) \neq \emptyset$. We only consider shapes with at least one edge, which ensures that $k_{\max}$ exists.

An algorithm solves the *shape containment problem instance* (S, A) if it determines whether $k = k_{\max}(S, A) = 0$ or $k > 0$ and, if $k > 0$, for every $r \in \{0, \ldots, 5\}$, every amoebot knows whether it is in $\mathcal{V}(k \cdot S^{(r)}, A)$ (see Fig. 3). We say that the algorithm solves the *shape containment problem* for S if it solves the problem instances (S, A) for all amoebot structures A. The algorithm may use a translated version of S as long as it contains the origin.

We will first present a lower bound of $\Omega(\sqrt{n})$ rounds for computing $\mathcal{V}(k \cdot S, A)$ that holds for an example shape S. Then, we combine several efficient primitives avoiding this lower bound to obtain the class of *snowflake shapes*, for which we solve the shape containment problem in $\mathcal{O}(\sqrt{n} \log n)$ rounds. Finally, we identify a subset of shapes for which $k_{\max}$ can be determined with a binary search, leading to an $\mathcal{O}(\log^2 k_{\max}(S, A))$ solution.

5 Related Work

The authors of [8] introduced the reconfigurable circuit extension with algorithms solving the leader election, compass alignment, and chirality agreement problems within $\mathcal{O}(\log n)$ rounds, w.h.p.[1] They also presented efficient solutions for some exact shape recognition problems. An amoebot structure can determine whether it matches a scaled version of a given shape composed of edge-connected faces in $\mathcal{O}(1)$ rounds. Further, parallelograms with linear or polynomial side ratios can be detected in $\Theta(\log n)$ rounds, w.h.p.

The *PASC algorithm* was introduced in [8, 12] and plays a crucial role for the results of this paper. It allows the computation of distances and was used to construct spanning trees and detect symmetry in polylogarithmic time, w.h.p. [12]. The authors in [11] used it to solve the shortest path forest problem, requiring $\mathcal{O}(\log \ell)$ rounds for a single source with ℓ destinations and $\mathcal{O}(\log n \log^2 k)$ rounds for k sources with any number of destinations.

In the context of computational geometry, the basic polygon containment problem was studied in [4], focusing on the case where only translation and rotation are allowed. The

[1] An event holds *with high probability* (w.h.p.) if it holds with probability at least $1 - n^{-c}$, where the constant c can be made arbitrarily large.

problem of finding the largest copy of a convex polygon inside some other polygon was discussed in [13] and [1], for example. Perhaps more closely related to our setting (albeit centralized) is an algorithm that solves the problem of finding the largest area parallelogram inside of an object in the triangular grid, where the object is a set of edge-connected faces [2].

6 Results

For the lower bound, we can show that for some shape S (see Fig. 4 (g)) and every amoebot algorithm $\mathcal{A}$ that terminates after $\mathrm{o}\left(\sqrt{n}\right)$ rounds, there exists an amoebot structure A for which $\mathcal{A}$ does not compute $\mathcal{V}(k_{\max}(S,A) \cdot S, A)$, even if $k_{\max}(S,A)$ is already known. This works because for shape 4 (g), we can construct many similar amoebot structures with different solution sets while forcing the relevant placement information to travel through a bottleneck shared by the structures. This bottleneck prevents any algorithm from distinguishing between the solution sets within $\mathrm{o}\left(\sqrt{n}\right)$ rounds.

For the upper bounds, we use the fact that amoebots can establish distributed *binary counters* to store and manipulate numbers by storing one bit in each amoebot along a path [5, 12]. The amoebots can then perform basic arithmetic operations with straightforward implementations of the standard written algorithms using circuits. Most of our primitives are applications of the PASC algorithm (see [8, 12]) combined with binary counters. The basic operation is to run the PASC algorithm on a straight line of amoebots, which allows each amoebot to compute its distance d to the start of the line bit by bit. Simultaneously, the bits of a value ℓ stored in a binary counter are transmitted on a global circuit, allowing each amoebot to compare d to ℓ. This way, we can identify lines of length ℓ in A. By manipulating ℓ and choosing appropriate lines to run the procedure, we can realize the following operations.

Let $\mathrm{L}(d,\ell)$ denote the *line* shape consisting of ℓ consecutive edges in some direction $d \in \mathcal{D}$ and let $\mathrm{T}(d,\ell)$ denote the *triangle* with side length ℓ that is spanned by $\mathrm{L}(d,\ell)$ and $\mathrm{L}(d,\ell)^{(1)}$ and contains all enclosed faces. Given ℓ in a binary counter, the amoebots can find all valid placements of $\mathrm{L}(d,\ell)$ and $\mathrm{T}(d,\ell)$ within $\mathcal{O}\left(\log\min\{\ell,n\}\right)$ rounds.

We now introduce primitives for transforming valid placements of one shape into valid placements of another shape. Observe that $p \in A$ is a valid placement of $S_1 \cup S_2$ if and only if it is a valid placement of both S_1 and S_2. Thus, given the valid placements of two shapes, the amoebots immediately know the valid placements of their union. Next, consider the *Minkowski sum* of two shapes: $S_1 \oplus S_2 := \{p_1 + p_2 \mid p_1 \in S_1, p_2 \in S_2\}$. Applying this operation to a shape S and a line results in a "stretched" shape $S \oplus \mathrm{L}(d,\ell)$. Given the valid placements of S and the length ℓ in a binary counter, our *stretch primitive* computes the valid placements of $S \oplus \mathrm{L}(d,\ell)$ in $\mathcal{O}\left(\log\min\{\ell,n\}\right)$ rounds. Finally, the *shift primitive* allows us to move the origin of some shapes in a cardinal direction, while adding edges to maintain connectivity. Note that moving the origin changes the positions of valid placements in the amoebot structure and the outcome of the union operation with this shape. The additional edges are a result of the operation itself. Given the valid placements of S and ℓ in a binary counter, the amoebots find the valid placements of $(S + \ell \cdot \boldsymbol{u}_d) \cup \mathrm{L}(d,\ell)$ in $\mathcal{O}\left(\log\min\{\ell,n\}\right)$ rounds, where $\boldsymbol{u}_d$ denotes the unit vector in direction d. This always works if $S = S' \oplus \mathrm{L}(d,\ell')$ for some $\ell' \in \mathbb{N}$.

We define the class of *snowflake* shapes recursively as all shapes that can be constructed by applying the union, stretch, and shift primitives a finite number of times to lines, triangles, and other snowflakes (see Fig. 4). By construction, the valid placements of a snowflake can be computed by applying the primitives sequentially and storing the valid placements of the intermediate shapes. For any given snowflake S, there is a sequence of these primitives with

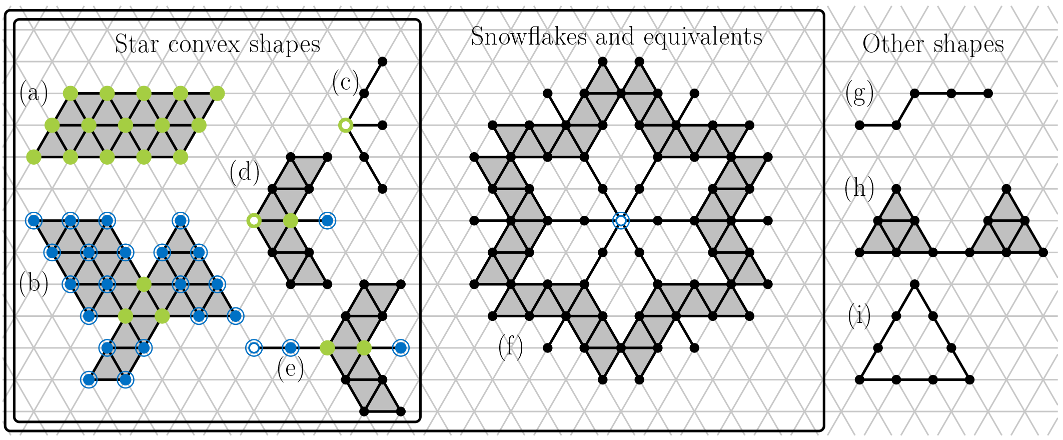


Figure 4 Examples of snowflakes, star convex shapes, and non-snowflake shapes. Green nodes are star convex shape centers, encircled blue nodes are possible snowflake origins, and origins used for operations have a white center. All center nodes are also snowflake origins. Shape (a) is convex and shape (b) demonstrates that not all snowflake origins must be center nodes. Shape (c) is a union of lines, (d) is the Minkowski sum of (c) and $\mathrm{L}(\mathrm{E}, 1)$, (e) is the result of shifting (d) by $2\boldsymbol{u}_{\mathrm{E}}$, and shape (f) is the union of six rotated variants of (e). (g) is the example shape for the lower bound.

constant length, which we encode in the amoebots' states to represent S. If a scale factor k is given in a binary counter, the amoebots take $\mathcal{O}(\log \min\{k, n\})$ rounds to find all valid placements of $k \cdot S$.

A shape S is *star convex* if it has no holes and contains a *center node* $c \in V(S)$ such that for every other node $v \in V(S)$, every shortest path from c to v in G_Δ is contained in S. We can show that every star convex shape is a snowflake (up to translation) and there is a surprising equivalent definition: A shape S is star convex if and only if for all scales $k, k' \in \mathbb{N}$ with $k < k'$, there exists a $t \in V_\Delta$ such that $k \cdot S + t \subseteq k' \cdot S$. Therefore, if S is star convex and there are no valid placements of $k \cdot S^{(r)}$ in A for any rotation r, then $k_{\max}(S, A) < k$. We use this fact to find the maximum scale of a star convex shape S by applying an *exponential search* to find an upper bound $K \leq 2 \cdot k_{\max}(S, A)$, followed by a *binary search* in $\{1, \ldots, K\}$. Thereby, we only construct the valid placements for $\mathcal{O}(\log k_{\max}(S, A))$ different scale factors, all of which have size $\mathcal{O}(k_{\max}(S, A))$. For snowflakes that are not star convex, we run a *linear search* instead, checking the scale factors $K, K-1, \ldots, k_{\max}(S, A)$ for some upper bound K computed beforehand. It turns out that for non-star convex snowflakes, we can use $K \leq k_{\max}(\mathrm{T}(d, 1), A) = \mathcal{O}(\sqrt{n})$. This leads to the main result of the paper:

▶ **Theorem 6.1.** *Let A be an amoebot structure and S be a snowflake shape. Programmed with a representation of S, the amoebots can compute $k = k_{\max}(S, A)$ in a binary counter and determine $\mathcal{V}(k \cdot S^{(r)}, A)$ for all $r \in \{0, \ldots, 5\}$ within $\mathcal{O}(\log^2 k)$ rounds if S is star convex and $\mathcal{O}(K \log K)$ rounds otherwise, where $k \leq K = k_{\max}(\mathrm{T}(E, 1), A) = \mathcal{O}(\sqrt{n})$.*

References

1 Pankaj K. Agarwal, Nina Amenta, and Micha Sharir. Largest Placement of One Convex Polygon Inside Another. *Discrete & Computational Geometry*, 19(1):95–104, 1998. `doi:10.1007/PL00009337`.

2 Md Abdul Aziz Al Aman, Raina Paul, Apurba Sarkar, and Arindam Biswas. Largest Area Parallelogram Inside a Digital Object in a Triangular Grid. In Reneta P. Barneva,

Valentin E. Brimkov, and Giorgio Nordo, editors, *Combinatorial Image Analysis - 21st International Workshop (IWCIA)*, volume 13348 of *LNCS*, pages 122–135, Cham, 2022. Springer. `doi:10.1007/978-3-031-23612-9_8`.

3 Ahmed Amine Chafik, Jaafar Gaber, Souad Tayane, Mohamed Ennaji, Julien Bourgeois, and Tarek El Ghazawi. From Conventional to Programmable Matter Systems: A Review of Design, Materials, and Technologies. *ACM Comput. Surv.*, 56(8):210:1–210:26, 2024. `doi:10.1145/3653671`.

4 Bernard Chazelle. The Polygon Containment Problem. *Advances in Computing Research*, 1(1):1–33, 1983.

5 Joshua J. Daymude, Robert Gmyr, Kristian Hinnenthal, Irina Kostitsyna, Christian Scheideler, and Andréa W. Richa. Convex Hull Formation for Programmable Matter. In Nandini Mukherjee and Sriram V. Pemmaraju, editors, *21st International Conference on Distributed Computing and Networking*, ICDCN ’20, pages 1–10, New York, NY, USA, 2020. ACM. `doi:10.1145/3369740.3372916`.

6 Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Brief Announcement: Amoebot - A New Model for Programmable Matter. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 220–222, Prague, Czech Republic, 2014. ACM. `doi:10.1145/2612669.2612712`.

7 Giuseppe A. Di Luna, Paola Flocchini, Nicola Santoro, Giovanni Viglietta, and Yukiko Yamauchi. Shape formation by programmable particles. *Distributed Computing*, 33(1):69–101, 2020. `doi:10.1007/s00446-019-00350-6`.

8 Michael Feldmann, Andreas Padalkin, Christian Scheideler, and Shlomi Dolev. Coordinating Amoebots via Reconfigurable Circuits. *Journal of Computational Biology*, 29(4):317–343, 2022. `doi:10.1089/cmb.2021.0363`.

9 Melvin Gauci, Radhika Nagpal, and Michael Rubenstein. Programmable Self-disassembly for Shape Formation in Large-Scale Robot Collectives. In Roderich Groß, Andreas Kolling, Spring Berman, Emilio Frazzoli, Alcherio Martinoli, Fumitoshi Matsuno, and Melvin Gauci, editors, *Distributed Autonomous Robotic Systems: The 13th International Symposium*, volume 6 of *Springer Proceedings in Advanced Robotics*, pages 573–586, Cham, 2018. Springer. `doi:10.1007/978-3-319-73008-0_40`.

10 Kyle W. Gilpin. *Shape Formation by Self-Disassembly in Programmable Matter Systems.* Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2012.

11 Andreas Padalkin and Christian Scheideler. Polylogarithmic Time Algorithms for Shortest Path Forests in Programmable Matter. In Ran Gelles, Dennis Olivetti, and Petr Kuznetsov, editors, *43rd ACM Symposium on Principles of Distributed Computing*, PODC ’24, pages 65–75, New York, NY, USA, 2024. ACM. `doi:10.1145/3662158.3662776`.

12 Andreas Padalkin, Christian Scheideler, and Daniel Warner. The Structural Power of Reconfigurable Circuits in the Amoebot Model. In Thomas E. Ouldridge and Shelley F. J. Wickham, editors, *28th International Conference on DNA Computing and Molecular Programming (DNA 28)*, volume 238 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:22, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.DNA.28.8`.

13 Micha Sharir and Sivan Toledo. Extremal polygon containment problems. *Computational Geometry*, 4(2):99–118, 1994. `doi:10.1016/0925-7721(94)90011-6`.

14 Pierre Thalamy, Benoît Piranda, and Julien Bourgeois. A survey of autonomous self-reconfiguration methods for robot-based programmable matter. *Robotics and Autonomous Systems*, 120:103242, 2019. `doi:10.1016/j.robot.2019.07.012`.

15 Tommaso Toffoli and Norman Margolus. Programmable Matter: Concepts and Realization. *International Journal of High Speed Computing*, 05(02):155–170, 1993. `doi:10.1142/S0129053393000086`.

16 Lidong Yang, Jiangfan Yu, Shihao Yang, Ben Wang, Bradley J. Nelson, and Li Zhang. A Survey on Swarm Microrobotics. *IEEE Transactions on Robotics*, 38(3):1531–1551, 2022. `doi:10.1109/TRO.2021.3111788`.

On plane cycles in geometric multipartite graphs

Marco Ricci, Jonathan Rollin, André Schulz, and Alexandra Weinberger

FernUniversität in Hagen, Germany
`firstname.lastname@fernuni-hagen.de`

Abstract

A geometric graph is a drawing of a graph in the plane where the vertices are drawn as points in general position and the edges as straight-line segments connecting their endpoints. It is plane if it contains no crossing edges. We study plane cycles in geometric complete multipartite graphs. We prove that if a geometric complete multipartite graph contains a plane cycle of length t, with $t \geq 6$, it also contains a smaller plane cycle of length at least $\lceil t/2 \rceil + 1$. We further give a characterization of geometric complete multipartite graphs that contain plane cycles in which at least two vertices are from the same partition class. For geometric drawings of $K_{n,n}$, we give a sufficient condition on the drawing under which they have, for each $s \leq n$, a plane cycle of length $2s$. Finally, we provide an algorithm that decides whether a given geometric drawing of $K_{n,n}$ contains a plane Hamiltonian cycle in time $O(n \log n + nk^2) + O(k^{5k})$, where k is the number of vertices inside the convex hull of all vertices.

1 Introduction

A *geometric graph* is a drawing of a graph G in the plane such that its vertices in $V(G)$ are drawn as disjoint points in general position (i.e., no three points lie on a line) and its edges in $E(G)$ are drawn as straight-line segments connecting their respective endpoints. We usually associate the vertices of the graph with their corresponding points in the plane and treat them interchangeable. A geometric graph is called *plane* if it contains no crossing edges. We also refer to a geometric graph whose abstract graph is G as a *drawing of* G. A geometric (host) graph H *contains* a plane graph G if there is a subdrawing of H without edge crossings, whose (abstract) graph is isomorphic to G.

Past research has mostly focused on questions where the host graph is a geometric complete graph. Cabello [6] showed that for a given graph G it is NP-hard to decide whether a geometric complete graph contains a plane G. However, if G is outerplanar, this can be decided in polynomial time; see for example Bose [5]. It is easy to see that each geometric complete graph H contains plane paths and cycles with up to $|V(H)|$ vertices. This is no longer true for other geometric host graphs. Figure 1 shows geometric complete multipartite graphs that do not contain plane cycles of certain lengths.

In this paper we consider geometric complete multipartite graphs as host graphs, with an emphasis on the bipartite case. We study whether these host graphs contain a plane cycle of a given length. We will usually refer to the vertex partition by its associated vertex coloring, where the color classes correspond to the partition classes. Throughout the paper we depict vertices from the same partition class by a common symbol/color in our figures. Moreover, we omit the edges of the host graph from our figures for a better readability, since they can be recovered from the vertex locations (symbols).

There has been much interest in related questions over the years, and collections of many of the obtained results are given in the surveys by Kaneko and Kano [10] and by Kano and Urrutia [13]. Particular attention has been paid to long plane paths and cycles. For geometric complete bipartite host graphs, Kaneko and Kano [11] showed that if one color

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.

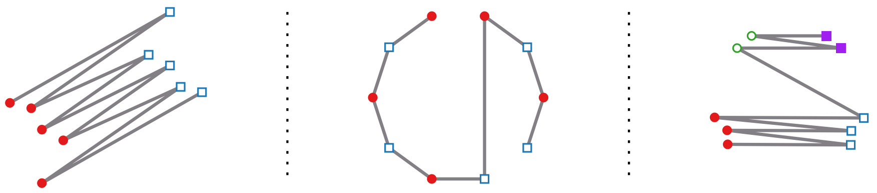


Figure 1 Plane paths in geometric complete multipartite graphs. Each graph contains a plane path on 10 vertices, but no plane 10-cycle. In fact, no plane cycle is contained in the graph on the left, the graph in the middle contains plane cycles of each even length at most 8, and the graph on the right contains exactly those plane cycles that visit at most one vertex from each color (called rainbow cycles).

class is sufficiently larger than the other, then any drawing contains a plane path containing all vertices of the smaller color class. Many known results focus on the special case of geometric complete bipartite host graphs with n vertices of each color where all vertices are in convex position. Mulzer and Valtr [17] showed that each such host graph contains a plane path of length at least $n + \epsilon n$, for some $\epsilon > 0$, improving previous results (cf. [14, 16]). For an upper bound in this case, constructions of such drawings where each plane path has length at most $\frac{4}{3}n + O(\sqrt{n})$ were given by Abellanas et al. [1] and Kynčl, Pach, and Tóth [14]. Other special (with respect to vertex placement) geometric graphs were investigated with regard to plane Hamiltonian paths and cycles by Cibulka et al. [7] and Soukup [18]. For geometric complete multipartite host graphs with an arbitrary number of color classes, Merino, Salazar and Urrutia [15] gave a lower bound on the length of the longest plane path contained in any such geometric graph and showed it is tight for odd numbers of colors. Weakening the noncrossing requirement, 1-plane Hamiltonian paths and cycles (i.e. paths and cycles of length $2n$) have been studied by Claverol et al. [8], and Hamiltonian paths and cycles with the minimal number of crossings have been studied by Kaneko et al. [12].

The algorithmic complexity of finding plane paths or cycles is open. However, Akitaya and Urrutia [3] and Abellanas et al. [1] described algorithms that decide whether a geometric complete bipartite host graph with all vertices in convex position has a plane Hamiltonian path in $O(n^2)$ time. Bandyapadhyay et al. [4] gave linear time algorithms to find plane Hamiltonian paths in drawings of complete bipartite graphs, where the vertices are mapped to the real line and the edges are drawn as circular arcs above and below the line.

We point out that a geometric drawing is uniquely determined by the vertex locations in the plane. Thus, the question whether a given geometric complete multipartite host graph H contains a plane G is equivalent to the question of whether the vertices of G can be mapped to colored points in the plane, given by $V(H)$, such that the endpoints of each edge in G are mapped to points of different colors and the resulting drawing of G is plane. Both perspectives are used in previous work. From another perspective, plane cycles can also be interpreted as polygons and there is a significant amount of research on so-called polygonizations (cf. [2]).

2 Monotonicity results for plane cycles

In this section, we are interested in the following "monotonicity" question: Does the existence of a plane cycle of length t in a geometric complete multipartite graph H imply the existence of a plane cycle of length t' in H for each t' with $3 \leq t' \leq t$? If the given plane cycle is one where all vertices have different colors (called a *rainbow cycle*), then this can be

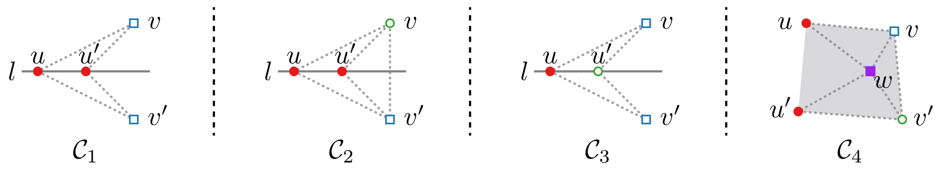


Figure 2 The configurations $\mathcal{C}_1 - \mathcal{C}_4$ which guarantee the existence of non-rainbow plane cycles. $\mathcal{C}_1 - \mathcal{C}_3$: Vertices v and v' lie on different sides of the straight line l through u and u'. $\mathcal{C}_4$: Vertices $\{u, u', v, v'\}$ form a convex quadrilateral with w in its interior.

trivially answered with yes. The question becomes interesting if at least two vertices of the given cycle have the same color; we call a cycle with this property *non-rainbow*. As complete bipartite graphs cannot contain any rainbow cycles, this is the only sensible case for the case of two colors (for which the question should only be asked for all t' even). But there are also geometric complete multipartite graphs — including bipartite ones — that do not contain any plane non-rainbow cycle; see Figure 1 (left and right) for examples. The next theorem characterizes those drawings. The characterization is based on four forbidden configurations $\mathcal{C}_1 - \mathcal{C}_4$, which are depicted in Figure 2. They are explained formally in a full version of this article where also the proofs for all results in this section are given.

▶ **Theorem 2.1.** *A geometric complete multipartite graph H contains a non-rainbow plane cycle if and only if it contains one of the configurations $\mathcal{C}_1 - \mathcal{C}_4$.*

There is an algorithm that checks whether H has a non-rainbow cycle in time $O(|V(H)|^5)$.

As $\mathcal{C}_1$ is the only forbidden configuration with only two colors, it follows that any geometric complete bipartite graph H contains a plane cycle if and only if it contains a plane cycle of length 4. With the help of Theorem 2.1 we can further easily check that the geometric graphs in Figure 1 (left and right) do not have non-rainbow plane cycles. Another consequence of Theorem 2.1 is that any geometric complete multipartite graph that has some non-rainbow plane cycle also has such a cycle of length 4 or 5. The following theorem gives a partial answer to the monotonicity question.

▶ **Theorem 2.2.** *Let H be a geometric complete multipartite graph which contains a non-rainbow plane cycle C of length t, where $t \geq 6$. Then H contains a shorter non-rainbow plane cycle of length at least $\lceil \frac{t}{2} \rceil + 1$.*

We note that — maybe counter-intuitively — it is not always possible to shorten an existing plane cycle by shortcutting a path with three edges; see Figure 3.

We conclude this section with a monotonicity result for a restricted type of drawing with n red and n blue vertices, which generalizes a result of Soukup [18, Remark after Theorem 1].

▶ **Theorem 2.3.** *Let H be a geometric complete bipartite graph with n vertices of each color. If there exists a set of blue vertices that contains in the interior of its convex hull all red vertices but no blue vertex, then for each $t \in \{2, \ldots, n\}$, H contains a plane cycle of length $2t$.*

3 An FPT algorithm with respect to the number of interior vertices

In this section we consider geometric complete bipartite host graphs H with n red vertices and n blue vertices. The vertices of H that lie in the interior of the convex hull of $V(H)$ are called the *interior vertices* and the remaining vertices are called *boundary vertices*. We sketch an algorithm that checks whether H contains a plane Hamiltonian cycle in time

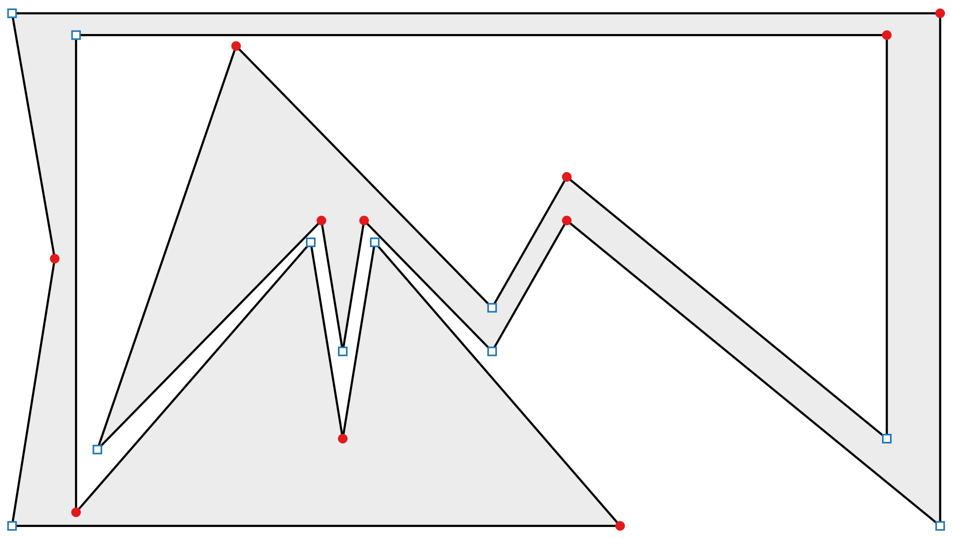


Figure 3 A plane cycle C of length 22 with no chord shortcutting C to a plane cycle of length 20.

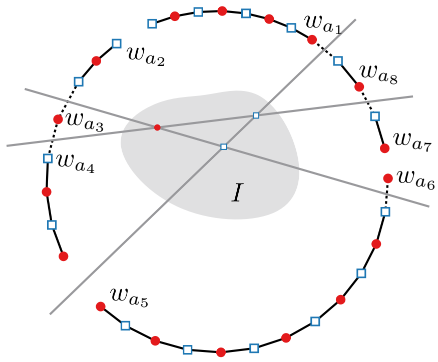


Figure 4 The vertices w_{a_2}, w_{a_5}, and w_{a_7} are critical vertices of the first kind, w_{a_1}, w_{a_3}, w_{a_4}, w_{a_6}, and w_{a_8} are critical vertices of the second kind. Critical arcs are shown by solid black edges.

$O(n \log n + nk^2) + O(k^{5k})$, where k is the number of interior vertices. A complete description is given in the full version of this article.

Let I denote the set of interior vertices and let $B = V(H) \setminus I$ denote the set of boundary vertices of H. Let $w_1, \ldots, w_m$ denote the boundary vertices in counterclockwise order where w_1 is chosen arbitrarily and indices are used modulo m (in particular, $w_0 = w_m$). The main idea is to split B into $O(k^2)$ parts, called critical arcs, such that it is sufficient to consider only few vertices within each part. A vertex $w_i \in B$ is *critical* if w_i and w_{i-1} are of the same color ("first kind") or there is a line through two vertices from I that separates w_i from w_{i-1} ("second kind"). Figure 4 shows an example. Let $S = \{w_{a_1}, \ldots, w_{a_s}\}$ denote the set of critical vertices where $a_1 < a_2 < \cdots < a_s$. For each critical vertex w_{a_i} the *critical arc at* a_i is the set $\{w_{a_i}, \ldots, w_{a_{i+1}-1}\}$ of boundary vertices between w_{a_i} and the next critical vertex (including the first and excluding the latter critical vertex). Observe that the critical arcs are pairwise disjoint and form a partition of B.

We now describe the desired algorithm. The idea is to iterate over all spanning cycles of the interior vertices I and check if one of them can be extended to the whole graph H.

An *initial cycle* F of I is a not-necessarily plane, geometric directed spanning cycle with vertex set I (that may use oriented edges from H as well as edges not in H), together with a nonempty subset G_F of the edges of F, called *gaps*, such that the edges of F not in G_F (called *fixed edges*) are in H and do not cross each other. The gaps represent those edges in F that should be replaced with longer paths using boundary vertices from B. In particular, every gap uv will be replaced by a path $u, w_i, w_{i+1}, \ldots, w_{i+j}, v$, where $w_i,\ w_{i+1}, \ldots, w_{i+j} \in B$. Our definition includes that in case $k = 2$, F is the directed cycle with just two edges between the two vertices in I. We call the endpoints of gaps in G_F *gap vertices*.

To extend F to B, for every gap vertex u we select a critical arc $A(u)$, and try to place a (new) neighbor for u in $A(u)$. The crucial observation is that we do not need to check all individual vertices from the selected critical arcs to decide whether F can be extended to B in this way. Instead, we find necessary and sufficient conditions for the selection of arcs alone to be "feasible". This gives a running time per selection that is independent of n as needed.

The final algorithm for the case $k \geq 2$ works as follows. We compute the critical arcs in $O(n \log n + nk^2)$ time by walking along the boundary vertices and check whether there are more than k critical vertices of the first kind. If not, we generate all $O(k!)$ directed spanning cycles of I. We then iterate, for every cycle, over all possible sets of gaps ($O(2^k)$ many) and check if this forms an initial cycle (fixed edges are noncrossing and in H) in $O(k^2)$ time for each. After some technical adjustments in linear time, we iterate over all $O(k^{4k})$ selections of critical arcs. For each selection we check "feasibility" in $O(k^2)$ time (a formal definition of feasibility is given in the full version). Altogether, this needs $O(n \log n + nk^2) + O(k^{5k})$ time, as desired. We conclude with the following theorem.

▶ **Theorem 3.1.** *There is an algorithm to check whether a geometric complete bipartite graph with k interior vertices contains a plane Hamiltonian cycle in $O(n \log n + nk^2) + O(k^{5k})$ time.*

4 Conclusion and open problems

We have shown that geometric complete multipartite graphs which contain a plane cycle C also admit a shorter plane cycle of at least half the size of C. For some restricted geometric complete bipartite graphs, we have shown that they contain plane cycles of each possible even length. This prompts the following question (distinguishing between two or more colors).

▶ **Open Problem 1.** *Let H be a geometric complete multipartite graph that contains a plane cycle of length t. Does H contain a plane cycle of length t' for any (even) t' with $3 \leq t' \leq t$?*

Further, we have given an FPT algorithm to decide whether a geometric complete bipartite graph contains a plane Hamiltonian cycle. It is still open to show whether this problem is NP-hard, which is related to an open question stated by Claverol et al. [8].

▶ **Open Problem 2.** *Is it NP-complete to decide if a given geometric complete multipartite graph contains a plane Hamiltonian cycle?*

The related problem of whether a set of edges of a geometric complete graph H can be completed to a plane Hamiltonian cycle in H is known to be NP-complete due to Akiyama et al. [2] and Jiang, Jiang, and Jiang [9]. In the full version of this article, we show that this problem is also NP-complete for geometric complete multipartite graphs.

References

1 Manuel Abellanas, Alfredo García, Ferran Hurtado, and Javier Tejel. Caminos alternantes. In *Proceedings of the X Encuentros de Geometría Computacional (EGC X)*, pages 7–12,

2003. In Spanish. English version available at https://dccg.upc.edu/people/ferran/alternatingPaths.pdf.

2 Hugo A. Akitaya, Matias Korman, Mikhail Rudoy, Diane L. Souvaine, and Csaba D. Tóth. Circumscribing polygons and polygonizations for disjoint line segments. In Gill Barequet and Yusu Wang, editors, *35th International Symposium on Computational Geometry (SoCG 2019)*, pages 9:1–9:17, 2019. doi:10.4230/LIPICS.SOCG.2019.9.

3 Jin Akiyama and Jorge Urrutia. Simple alternating path problem. *Discrete Mathematics*, 84(1):101–103, 1990. doi:10.1016/0012-365X(90)90276-N.

4 Sayan Bandyapadhyay, Aritra Banik, Sujoy Bhore, and Martin Nöllenburg. Geometric planar networks on bichromatic collinear points. *Theoretical Computer Science*, 895:124–136, 2021. doi:10.1016/J.TCS.2021.09.035.

5 Prosenjit Bose. On embedding an outer-planar graph in a point set. *Computational Geometry*, 23(3):303–312, 2002. doi:10.1016/S0925-7721(01)00069-4.

6 Sergio Cabello. Planar embeddability of the vertices of a graph using a fixed point set is np-hard. *Journal of Graph Algorithms and Applications*, 10(2):353–363, 2006. doi:10.7155/JGAA.00132.

7 Josef Cibulka, Jan Kynčl, Viola Mészáros, Rudolf Stolař, and Pavel Valtr. Hamiltonian alternating paths on bicolored double-chains. In Ioannis G. Tollis and Maurizio Patrignani, editors, *16th International Symposium Graph Drawing (GD 2008)*, pages 181–192. Springer, 2009. doi:10.1007/978-3-642-00219-9_18.

8 Mercè Claverol, Alfredo García Olaverri, Delia Garijo, Carlos Seara, and Javier Tejel. On Hamiltonian alternating cycles and paths. *Computational Geometry*, 68:146–166, 2018. doi:10.1016/J.COMGEO.2017.05.009.

9 Rain Jiang, Kai Jiang, and Minghui Jiang. Linking disjoint segments into a simple polygon is hard. Preprint, 2021. URL: http://arxiv.org/abs/2108.12812, arXiv:2108.12812.

10 Atsushi Kaneko and M. Kano. Discrete geometry on red and blue points in the plane — a survey —. In Boris Aronov, Saugata Basu, János Pach, and Micha Sharir, editors, *Discrete and computational geometry: The Goodman-Pollack Festschrift*, volume 25 of *Algorithms Combin.*, pages 551–570. Springer, 2003. doi:10.1007/978-3-642-55566-4_25.

11 Atsushi Kaneko and Mikio Kano. On paths in a complete bipartite geometric graph. In Jin Akiyama, Mikio Kano, and Masatsugu Urabe, editors, *Japanese Conference on Discrete and Computational Geometry (JCDCG 2000)*, pages 187–191, 2001. doi:10.1007/3-540-47738-1_17.

12 Atsushi Kaneko, Mikio Kano, and Kiyoshi Yoshimoto. Alternating Hamilton cycles with minimum number of crossings in the plane. *International Journal of Computational Geometry & Applications*, 10(1):73–78, 2000. doi:10.1142/S021819590000005X.

13 Mikio Kano and Jorge Urrutia. Discrete geometry on colored point sets in the plane—A survey. *Graphs and Combinatorics*, 37(1):1–53, 2021. doi:10.1007/S00373-020-02210-8.

14 Jan Kynčl, János Pach, and Géza Tóth. Long alternating paths in bicolored point sets. *Discrete Mathematics*, 308(19):4315–4321, 2008. doi:10.1016/j.disc.2007.08.013.

15 Criel Merino, Gelasio Salazar, and Jorge Urrutia. On the length of longest alternating paths for multicoloured point sets in convex position. *Discrete Mathematics*, 306(15):1791–1797, 2006. doi:10.1016/J.DISC.2006.03.035.

16 Viola Mészáros. *Extremal problems on planar point sets.* Phd thesis, University of Szeged, Bolyai Institute, 2011.

17 Wolfgang Mulzer and Pavel Valtr. Long alternating paths exist. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry (SoCG 2020)*, pages 57:1–57:16, 2020. doi:10.4230/LIPIcs.SoCG.2020.57.

18 Jan Soukup. Bicolored point sets admitting non-crossing alternating Hamiltonian paths. Preprint, 2024. URL: http://arxiv.org/abs/2404.06105, arXiv:2404.06105.

Property Testing of Curve Similarity

Peyman Afshani[1], Maike Buchin[2], Anne Driemel[3], Marena Richter[3], and Sampson Wong[4]

1 **Department of Computer Science, Aarhus University, Denmark**
2 **Faculty of Computer Science, Ruhr University Bochum, Germany**
3 **Institute for Computer Science, University of Bonn, Germany**
4 **Department of Computer Science, University of Copenhagen, Denmark**

Abstract

We propose a sublinear algorithm for probabilistic testing of the discrete Fréchet distance. We assume the algorithm is given access to the input curves via a query oracle that returns the set of vertices of the curve that lie within a radius δ of a specified vertex of the other curve. The goal is to use a small number of queries to determine with constant probability whether the two curves have discrete Fréchet distance at most δ or they are "ε-far" (for $0 < \varepsilon < 2$) from being similar, i.e., an ε-fraction of the curves must be ignored for them to become similar. We present an algorithm that is sublinear under two assumptions (i) that the curves are κ-straight, meaning they are κ-approximate shortest paths in the ambient metric space, for some $\kappa \ll n$, and (ii) that they have edge lengths within a constant factor of δ (we later relax this toward a weaker uniform sampling condition). The algorithm uses $O(\frac{\kappa}{\varepsilon} \log \frac{\kappa}{\varepsilon})$ queries and it is given the value of κ in advance. Our algorithm works in a matrix representation of the input and may be of independent interest to matrix testing.

Related Version arXiv:2502.17277s

1 Introduction

We initiate a study of property testing for measures of curve similarity, motivated by the need for fast solutions for curve classification and clustering. We believe that property testing for well-studied measures like the Fréchet distance is especially well motivated due to its connections to other problems studied on the curves, such as clustering [5, 10], similarity search [3, 9, 16] and map reconstruction [6, 7].

Typically in property testing, we are given access to a (large) data set and the goal is to very quickly assess whether the data has a certain property. Instead of the classical notation of correctness, a property testing algorithm is considered correct if it can satisfy the following two conditions, with a probability close to 1: First, if the input has the desired property, the algorithm must return *accept* and second, if the input is "far" from having the property (under some suitable definition of "far"), the algorithm should *reject* the input. For more information, details and motivation on property testing, see [4, 25].

Property testing algorithms can for example be useful if the input is extremely large. Then, it makes sense to obtain a quick approximate answer before deciding to run more expensive algorithms. For the Fréchet distance, there are applications [3, 9, 16] where negative filters are used to minimize expensive Fréchet distance computations. Another motivation for property testing is when small errors can be tolerated or objects that are close to having the desired property are acceptable. In fact, our chosen error model is very close to a partial Fréchet distance [8]. For more details on motivations to study property testing, see [19, 25].

Computational geometry has a long tradition of using randomization and sampling to speed up algorithmic approaches [20, 21, 22, 24]. Property testing has received some attention within computational geometry, but is much less explored compared to other areas. There are fast and efficient testers for many basic geometric properties, such as

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

convexity of a point set [15], disjointness of objects [15], the weight of Euclidean minimum spanning tree [12, 13, 14], clustering [23], the Voronoi property of a triangulation [15], ray shooting [11, 15], as well as LP-type problems [17].

We mention that there is also work on matrix testing [18]. Compared to our setting, the input size is defined as n^2—the number of entries needed to specify the matrix. In our case, we assume the input size to be n, the maximum of column and row dimension of the matrix.

We believe that a more general variant of our tester can be easily adapted to test the continuous Fréchet distance [1] by subsampling vertices along the edges of the curves, as the dependency of the query complexity is independent of the number of such samples n.

2 Preliminaries

Let (M, d) be a metric space. We say a *(discrete) curve* P in (M, d) is an ordered point sequence $\langle p_1, \dots, p_n \rangle$ with $p_i \in M$ for all $i = 1, \dots, n$. We call the points of the curve *vertices*. We denote by $|P|$ the number of vertices in P and by $\ell(P)$ its *length*, which is defined as $\ell(P) = \sum_{i=1}^{n-1} d(p_i, p_{i+1})$. The *subcurve* of P between p_i and p_j is denoted by $P[i, j]$. A curve P is called *κ-straight* if for any two vertices p_i and p_j in P, we have $\ell(P[i, j]) \leq \kappa \cdot d(p_i, p_j)$. Let $[n]$ be the set of integers from 1 to n and by $[n] \times [n] \subset \mathbb{N} \times \mathbb{N}$ the integer lattice of n times n integers. Given two curves $P = \langle p_1, \dots, p_n \rangle$ and $Q = \langle q_1, \dots, q_n \rangle$, we say that an ordered sequence $\mathcal{C}$ of elements in $[n] \times [n]$ is a *coupling* of P and Q, if it starts in $(1, 1)$, ends in (n, n) and for any consecutive tuples $(i, j), (i', j')$ in $\mathcal{C}$ it holds that $(i', j') \in \{(i + 1, j), (i, j + 1)\}$. We define the *discrete Fréchet distance*[1] between P and Q as follows

$$D_{\mathcal{F}}(P, Q) := \min_{\text{coupling } \mathcal{C}} \max_{(i,j) \in \mathcal{C}} d(p_i, q_j).$$

For brevity, we simply call this the Fréchet distance between P and Q. One can verify that the Fréchet distance satisfies the triangle inequality. The *free space matrix* of P and Q with distance value δ is an $n \times n$ matrix M_δ, where the i-th column corresponds to the vertex p_i of P and the j-th row corresponds to the vertex q_j of Q.

The entry $M_\delta[i, j]$ has the value 0 if $d(p_i, q_j) \leq \delta$ and 1 otherwise.[2] A coupling $\mathcal{C}$ is a path through the free space matrix that always moves only one step up or one step to the right. We call these paths *monotone Manhattan paths*. We define the *cost* of such a path as $c(\mathcal{C}) = \sum_{(i,j) \in \mathcal{C}} M_\delta[i, j]$. Note that the Fréchet distance between P and Q is at most δ if and only if there exists a monotone Manhattan path with cost 0 from $(1, 1)$ to (n, n).

Our analysis is based on a property of the free space matrix. We first define this property and then link the property to a certain class of well-behaved input curves.

▶ **Definition 2.1** (t-local)**.** Let M be a free space matrix of curves P and Q. We say that M is *t-local* if, for any tuples (i_1, j_1) and (i_2, j_2) with $M[i_1, j_1] = 0 = M[i_2, j_2]$, it holds that $|i_1 - i_2| \leq t \cdot (2 + |j_1 - j_2|)$ and $|j_1 - j_2| \leq t \cdot (2 + |i_1 - i_2|)$.

▶ **Observation 2.2.** *Suppose M is t-local. If we have $M[i, j] = 0 = M[i, j']$, then we have $|j - j'| \leq 2t$. If we have $M[i, j] = 0 = M[i', j]$, we have $|i - i'| \leq 2t$.*

[1] The classical definition of the discrete Fréchet distance allows diagonal steps in the coupling. We follow the definition used in [2] which does not allow diagonal steps as it simplifies our proofs. An easy adaptation of our proofs to a definition with the diagonal steps can be found in the full version.

[2] Note we use 0 and 1 in switched roles compared to the conventions in the literature on the free space matrix. Our notation makes the cost-function of the path more intuitive.

q_7	1	1	1	1	1	**0**	**0**
	1	1	1	1	**0**	**0**	1
	1	1	1	1	**0**	1	1
	1	1	1	1	**1**	1	0
	1	**0**	**0**	**1**	**1**	0	0
	0	**0**	1	1	0	0	1
q_1	**0**	1	1	0	0	1	1
	p_1						p_7

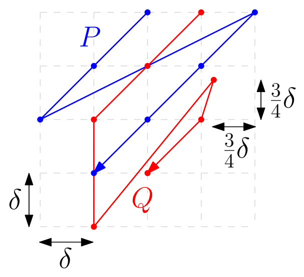


Figure 1 On the left we have M_δ for the curves on the right. The gray path is a minimum cost monotone Manhattan path of cost 3.

Lemma 2.3 below shows that the free space matrix of κ-straight curves is $\mathcal{O}(\kappa)$-local. For simplicity, we assume that the lengths of the edges are bounded by a fixed multiple of δ, the query radius of the Fréchet distance. In the full version, we show how to relax this assumption with little overhead. For a proof of this lemma, we refer to the full version.

▶ **Lemma 2.3.** *Let P and Q be κ-straight with edge lengths in $[\delta/\alpha, \alpha\delta]$ for some constant $\alpha \geq 1$ and let M be their free space matrix with distance value δ. Then, M is $\mathcal{O}(\kappa)$-local.*

3 Property testing

The problem we study in this paper is the following. Assume we want to determine for two curves P and Q, each consisting of n vertices,[3] if their Fréchet distance is at most a given value of δ. We do not have direct access to the input curves. Instead, we have access to an oracle that returns the information in a given row or column of the δ-free space matrix in the form of a sorted list of indices of zero-entries. We call this a *query* and we want to determine $D_{\mathcal{F}}(P, Q) \leq \delta$ using as few (sublinear in n) queries as possible. Note that from the point of view of a data structure setting, our query corresponds to a classical ball range query with a vertex p of one curve and returns the list of vertex indices of the other curve that are contained in the ball of radius δ centered at p.

Our bounds on the number of queries will be probabilistic and hold under a certain error model. The error model allows for the coupling path to pass through a bounded number of one-entries of the free space matrix.

▶ **Definition 3.1** ((ε, δ)-far)**.** Given two curves P and Q consisting of n vertices each, we say that P and Q are (ε, δ)-far from each other if there exists no monotone Manhattan path from $(1, 1)$ to (n, n) in the δ-free space matrix of cost εn or less.

▶ **Definition 3.2** (Fréchet-tester)**.** Assume we are given query-access to two curves P and Q, and we are given values $\delta > 0$ and $0 < \varepsilon < 2$. If the two curves have Fréchet distance at most δ, we must return "yes", and if they are (ε, δ)-far from each other w.r.t. the Fréchet distance, the algorithm must return "no", with probability at least $\frac{4}{5}$.

[3] For ease of notation, our analysis assumes the input curves have the same number of vertices.

4 Testing the Fréchet distance

The idea of Algorithm 1 is to sample a number of columns and rows and check whether there locally exists a monotone Manhattan path of cost zero. The following definition classifies when such a (local) path exists.

▶ **Definition 4.1** (Permeability). We say a block $[i, i']$ of consecutive columns (resp., rows) from index i to index i' is *permeable* if there exists a monotone Manhattan path of cost zero that starts in column (resp., row) i and ends in column (resp., row) i'.

If a column (resp. row) is individually not permeable, i.e. it contains only one-entries, we call it a *barrier-column* (resp. *barrier-row*). Note that any non-permeable block of rows or columns is a witness to the fact that no global monotone Manhattan path of cost 0 exists and therefore the curves are not at distance δ to each other. In that case, the algorithm can answer "no".

▶ **Definition 4.2** (Permeability query). To check if a block of k columns (or rows) is permeable, the algorithm first performs k queries to obtain the positions of zero-entries in these columns (or rows), then we build the induced subgraph of the grid for these zero-entries, connect all neighboring zero-entries according to the possible steps of a coupling and then connect all zero-entries of the last column to a sink and all zero-entries of the first column from a source. It remains to check if there is a path from source to sink, which can be done in linear time in the total number of zero-entries queried since the graph is acyclic.

Algorithm 1 Fréchet-Tester(M, t, ε)

```
1.  If $M[1,1] = 1$ or $M[n,n] = 1$ then return "no".
2.  repeat $\lceil \frac{24t}{\varepsilon} \rceil$ times:
3.      $j \leftarrow$ sample an index uniformly at random from $[n]$.
4.      if row $j$ or column $j$ of $M$ is a barrier-row or -column then return "no".
5.  $K \leftarrow \lceil \frac{\varepsilon n}{32t} \rceil - 1$, $\ell \leftarrow \lceil \frac{128t}{\varepsilon} \rceil$, let $\mathcal{J}$ be a set of intervals and set $\mathcal{J} \leftarrow \emptyset$.
6.  for $i = 0, \ldots, \lfloor \log_2 \ell \rfloor$ do:
7.      $I \leftarrow$ sample $\lceil \frac{16n}{2^{i+1}K} \rceil$ different indices uniformly at random from $\{0, 1, \ldots, \frac{n}{2^{i+1}} - 2\}$.
8.      for each $j \in I$ do: add $[j2^{i+1}, (j+2)2^{i+1}]$ to $\mathcal{J}$.
9.  foreach $[i, j] \in \mathcal{J}$ do:
10.     if block $[i, j]$ of consecutive columns is not permeable then return "no".
11.     if block $[i, j]$ of consecutive rows is not permeable then return "no".
12. return "yes".
```

Our analysis is based on two lemmas. For all proofs in this section, we refer to the full version. The first lemma shows that if an optimal path contains a long sequence of one-entries, there must be many barrier-columns and -rows.

▶ **Lemma 4.3** (Barrier Lemma). *Let M be t-local and let $\mathcal{C}$ be a monotone Manhattan path of minimum cost. Suppose that there are two zero-entries $(i, j), (i', j') \in \mathcal{C}$ such that $\mathcal{C}$ visits no zero-entry and at least $4t$ one-entries in between them. Then there is a total of at least $\lceil \frac{i'-i+j'-j}{2t} \rceil$ barrier-rows between j and j' and barrier-columns between i and i'.*

The second lemma shows that if an optimal path has a long stretch with relatively many one-entries on the path, there cannot be a long monotone Manhattan path of zero-entries in the same sequence of rows or columns of the matrix M, implying impermeability.

▶ **Lemma 4.4** (Impermeability Lemma). *Let M be t-local. Let $\mathcal{C}$ be a monotone Manhattan path through M of minimum cost. Suppose $(i,j),(i',j') \in \mathcal{C}$ with $j'-j > 2t$ (resp. $i'-i > 2t$) correspond to zero-entries in M and the subpath of $\mathcal{C}$ from (i,j) to (i',j') visits at least $4t-1$ one-entries. Then, the block of columns $[i,i']$ (resp. rows $[j,j']$) is not permeable.*

▶ **Lemma 4.5** (Main Lemma). *Let M be t-local and $M[1,1] = 0 = M[n,n]$. Suppose the total number of barrier-rows and -columns is at most $\frac{\varepsilon n}{8t}$. Let $\mathcal{C}$ be a monotone Manhattan path with lowest cost through M and suppose that $c(\mathcal{C}) > \varepsilon n$. Then, with probability at least $\frac{9}{10}$ at least one sampled interval in the set $\mathcal{J}$ during Algorithm 1 corresponds to a non-permeable block of columns or rows.*

Proof Sketch. We decompose the path $\mathcal{C}$ into a minimum number of groups that each contain more than $4t$ one-entries and start and end with a zero-entry. Using the barrier lemma, we can show that at most $\frac{\varepsilon n}{2}$ one-entries of $\mathcal{C}$ are in groups that visit more than $8t$ one-entries. So at least $\frac{\varepsilon n}{2}$ one-entries of $\mathcal{C}$ are in groups with at most $8t$ one-entries per group. We can argue that at least half of these groups have height and width at most $\frac{128t}{\varepsilon}$ and either the height or the width is larger than $2t$. So for each of these small groups we can apply the Impermeability Lemma to the rows or to the columns, meaning that a Permeability query on this group fails. Finally, we can show that one of these intervals will be included in one of the intervals chosen in Line 8 with sufficiently high probability. ◀

▶ **Theorem 4.6.** *Let P and Q be curves with n vertices such that their free space matrix is t-local and t is known. Then, Algorithm 1 is a Fréchet-tester that needs $\mathcal{O}(\frac{t}{\varepsilon}\log\frac{t}{\varepsilon})$ queries.*

Proof Sketch. Consider the case that a minimum cost monotone Manhattan path visists more than εn one-entries. We can show that the algorithm returns "no" in Line 4 with probability at least $\frac{9}{10}$ if there are at least $\frac{\varepsilon n}{8t}$ barrier-columns and -rows. If there are less than $\frac{\varepsilon n}{8t}$ barrier-columns and -rows, the Lemma 4.5 yields that we will return "no" with probability at least $\frac{9}{10}$. We can show that the algorithm performs $\mathcal{O}(\frac{t}{\varepsilon} + \frac{n \log \ell}{K})$ queries. If we plug in the values of K and ℓ, this yields the desired number of queries. ◀

References

1 Helmut Alt, Christian Knauer, and Carola Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38:45–58, 2004.

2 Rinat Ben Avraham, Omrit Filtser, Haim Kaplan, Matthew J Katz, and Micha Sharir. The discrete Fréchet distance with shortcuts via approximate distance counting and selection. In *Proceedings of the thirtieth annual symposium on computational geometry*, pages 377–386, 2014.

3 Julian Baldus and Karl Bringmann. A fast implementation of near neighbors queries for Fréchet distance (GIS cup). In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 99:1–99:4. ACM, 2017. doi:10.1145/3139958.3140062.

4 Arnab Bhattacharyya and Yuichi Yoshida. *Property Testing: Problems and Techniques*. Springer Nature, 2022.

5 Milutin Brankovic, Kevin Buchin, Koen Klaren, André Nusser, Aleksandr Popov, and Sampson Wong. (k, ℓ)-medians clustering of trajectories using continuous dynamic time warping. In *SIGSPATIAL '20: 28th International Conference on Advances in Geographic Information Systems*, pages 99–110. ACM, 2020. doi:10.1145/3397536.3422245.

6 Kevin Buchin, Maike Buchin, David Duran, Brittany Terese Fasy, Roel Jacobs, Vera Sacristán, Rodrigo I. Silveira, Frank Staals, and Carola Wenk. Clustering trajectories for

map construction. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 14:1–14:10. ACM, 2017. doi:10.1145/3139958.3139964.

7 Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Jorren Hendriks, Erfan Hosseini Sereshgi, Vera Sacristán, Rodrigo I. Silveira, Jorrick Sleijster, Frank Staals, and Carola Wenk. Improved map construction using subtrajectory clustering. In *LocalRec'20: Proceedings of the 4th ACM SIGSPATIAL Workshop on Location-Based Recommendations, Geosocial Networks, and Geoadvertising*, pages 5:1–5:4. ACM, 2020. doi:10.1145/3423334.3431451.

8 Kevin Buchin, Maike Buchin, and Yusu Wang. Exact algorithms for partial curve matching via the Fréchet distance. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 645–654. SIAM, 2009.

9 Kevin Buchin, Yago Diez, Tom van Diggelen, and Wouter Meulemans. Efficient trajectory queries under the Fréchet distance (GIS cup). In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 101:1–101:4. ACM, 2017. doi:10.1145/3139958.3140064.

10 Kevin Buchin, Anne Driemel, Natasja van de L'Isle, and André Nusser. klcluster: Center-based clustering of trajectories. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 496–499. ACM, 2019. doi:10.1145/3347146.3359111.

11 Bernard Chazelle, Ding Liu, and Avner Magen. Sublinear geometric algorithms. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 531–540, 2003.

12 Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM Journal on Computing*, 34(6):1370–1379, 2005. doi:10.1137/S0097539702403244.

13 Artur Czumaj, Funda Ergün, Lance Fortnow, Avner Magen, Ilan Newman, Ronitt Rubinfeld, and Christian Sohler. Approximating the weight of the Euclidean minimum spanning tree in sublinear time. *SIAM Journal on Computing*, 35(1):91–109, 2005. doi:10.1137/S0097539703435297.

14 Artur Czumaj and Christian Sohler. Estimating the weight of metric minimum spanning trees in sublinear time. *SIAM Journal on Computing*, 39(3):904–922, 2009. doi:10.1137/060672121.

15 Artur Czumaj, Christian Sohler, and Martin Ziegler. Property testing in computational geometry. In *European Symposium on Algorithms*, pages 155–166. Springer, 2000.

16 Fabian Dütsch and Jan Vahrenhold. A filter-and-refinement-algorithm for range queries based on the Fréchet distance (GIS cup). In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 100:1–100:4. ACM, 2017. doi:10.1145/3139958.3140063.

17 Rogers Epstein and Sandeep Silwal. Property testing of lp-type problems. In *47th International Colloquium on Automata, Languages, and Programming*, volume 168 of *LIPIcs*, pages 98:1–98:18, 2020. doi:10.4230/LIPICS.ICALP.2020.98.

18 Eldar Fischer and Ilan Newman. Testing of matrix properties. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 286–295, 2001.

19 Oded Goldreich. *Introduction to property testing.* Cambridge University Press, 2017.

20 J.E. Goodman, J. O'Rourke, and C.D. Tóth. *Handbook of discrete and computational geometry, third edition.* 01 2017. doi:10.1201/9781315119601.

21 Sariel Har-Peled. *Geometric Approximation Algorithms.* American Mathematical Society, USA, 2011.

22 Jirí Matousek. *Lectures on discrete geometry*, volume 212 of *Graduate texts in mathematics.* Springer, 2002.

23 Morteza Monemizadeh. Facility Location in the Sublinear Geometric Model. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*, volume 275 of *LIPIcs*, pages 6:1–6:24, 2023. `doi:10.4230/LIPIcs.APPROX/RANDOM.2023.6`.

24 Ketan. Mulmuley. *Computational geometry : an introduction through randomized algorithms*. Prentice-Hall, Englewood Cliffs, N.J, 1994.

25 Dana Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends® in Theoretical Computer Science*, 5(2):73–205, 2010.

Finding Complex Patterns in Trajectory Data via Geometric Set Cover*

Jacobus Conradi[1] **and Anne Driemel**[1]

1 **Department of Computer Science, University of Bonn**

Abstract

Clustering trajectories is a central challenge when confronted with large amounts of movement data such as GPS data. We study a clustering problem that can be stated as a geometric set cover problem: Given a polygonal curve of complexity n, find the smallest number k of representative trajectories, each of complexity at most l, such that any point on the input trajectory lies on a subtrajectory of the input that has Fréchet distance at most Δ to one of the representative trajectories. A previous work by Brüning et al. (2022) presented a bicriteria approximation algorithm that returns a set of curves of size $O(kl\log(kl))$ which covers the input with a radius of 11Δ in time $\tilde{O}((kl)^2n+kln^3)$, where k is the smallest number of curves of complexity l needed to cover the input with a distance of Δ. The representative trajectories computed by their algorithm are always line segments, while other known algorithms do not have any output guarantees. In applications however, one is usually interested in representative curves which consist of several edges. We present a new approach that builds upon previous work computing a set of curves of size $O(k\log(n))$ in time $\tilde{O}(kn^3)$ with the same distance guarantee of 11Δ, where each curve may consist of curves of complexity up to the given complexity parameter l. We conduct experiments on tracking data of ocean currents and full body motion data suggesting its validity as a tool for analyzing large spatio-temporal data sets.

Related Version The full version can be accessed at `https://arxiv.org/abs/2308.14865`.

1 Introduction

Advancements in motion tracking technology made it possible to track motion data from many different areas affected by climate change ranging from ocean currents to animal migration. [19] observed changes in ocean currents driven by climate change and analysed how these affect the dispersal of marine life providing evidence of the importance of understanding and predicting these current changes. Similar effects can be observed among migratory land-bound animals, as temperature and resource availability changes [10] and displacement of human life [16]. Practitioners are often confronted with vast amounts of data from which one would like to extract a reoccuring pattern, preferably of small complexity, making the data more accessible to less efficient algorithms or schematic visualization.

Identifying such patterns is a particular challenge, which is the motivation for algorithms based on heuristics [11] or reinforcement learning [12]. As the type of pattern which is sought after may vary depending on the data, and thus the quality measurement for a certain pattern varies, there are multiple approaches to tackle this problem (see the survey papers

* Partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 313421352 (FOR 2535 Anticipating Human Behavior) and the iBehave Network: Sponsored by the Ministry of Culture and Science of the State of North Rhine-Westphalia. The authors are affiliated with Lamarr Institute for Machine Learning and Artificial Intelligence. We thank Frederik Brüning for contributions in early stages of this research. We thank Jürgen Gall, Julian Tanke, Jürgen Kusche, and Bernd Uebbing for useful discussions on the data sets and real world problems. Special thanks to Simon Bartlmae and Paul Jünger for their assistance in conducting the experiments.

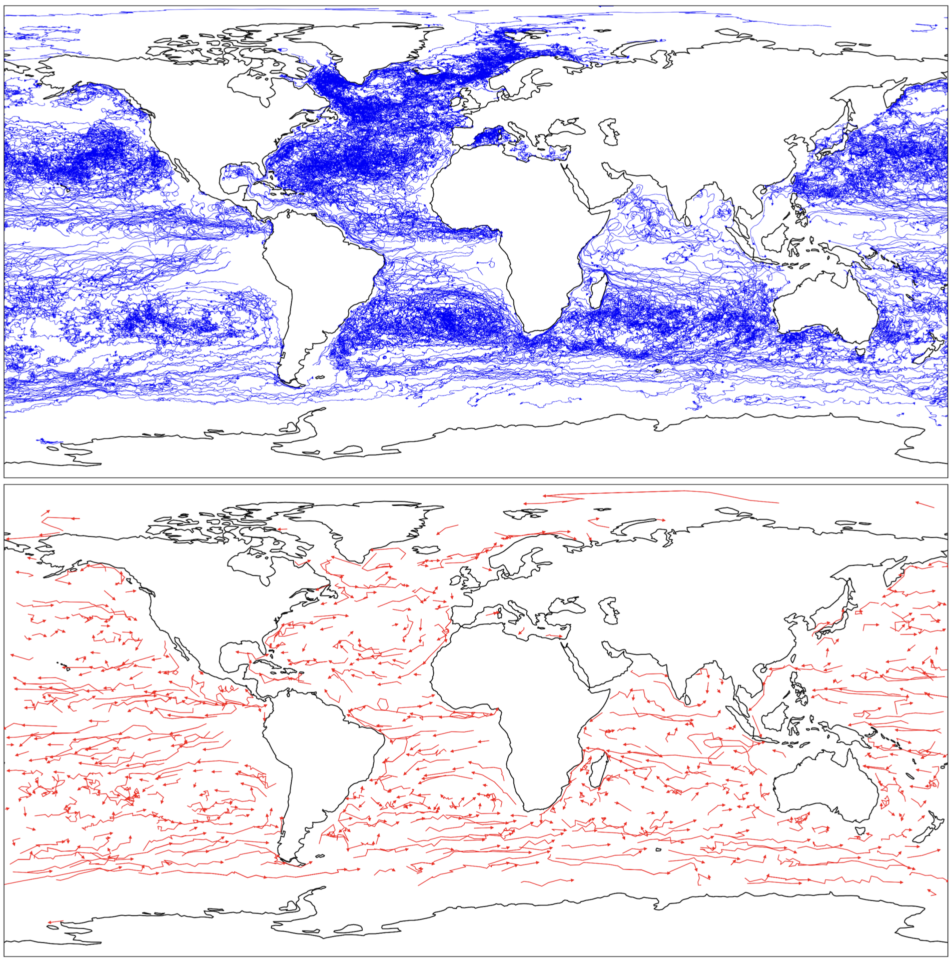


Figure 1 Illustration of ≈ 2000 individual ocean surface drifters and a resulting clustering.

[5, 18, 20]). One quality measure often used in this context is that of the Fréchet distance as a measure of similarity between the given trajectories. Examples are the work of Agarwal et al. [1] or that of Buchin et al. [4]. The approach we want to focus on is that of Akitaya et al. [3] who pose the problem as a geometric set cover problem, in which a given trajectory needs to be "covered" by the smallest possible number of "center" trajectories such that each point of the input trajectory is contained in a subtrajectory of the input trajectory which has a small Fréchet distance to one of the center trajectories. This can similarly be thought of as a clustering problem in which each point on the input trajectory is assigned to at least one center trajectory. One drawback for practical applications of the approaches presented in [3] and the subsequent work of Brüning et al. [2] is that the center trajectories computed only ever consist of a single edge, whereas in applications one is often interested in finding center trajectories of higher complexity, as even a simple circular motion (as is present in gulf streams for example) cannot be modeled well with a single edge. In this paper we focus on extending the approach of [2] to center trajectories of higher complexity and analyzing our approach on real data.

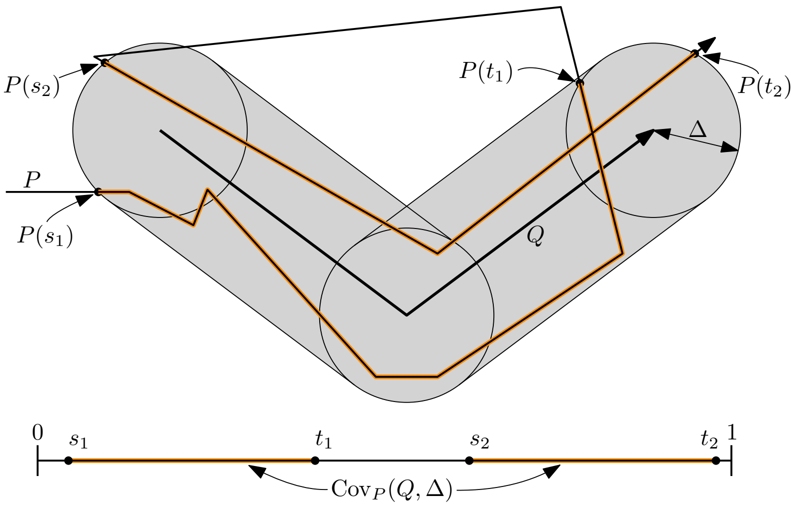


Figure 2 Illustration of the Δ-coverage of Q on the curve P.

2 Previous Work

We consider a given polygonal curve P parametrized over $[0,1]$ together with a distance threshold $\Delta > 0$ and complexity bound $l \in \mathbb{N}_{\geq 2}$. The problem is to find a set of curves $\mathcal{C} = \{C_1, \dots, C_k\}$, each of complexity at most l, such that every point $P(p)$ for $p \in [0,1]$ is contained in a subcurve π_p of P and there is a curve $C_i \in \mathcal{C}$ that has continuous Fréchet distance $\mathrm{d}_{\mathcal{F}}(\pi_p, C_i) \leq \Delta$. The continuous ***Fréchet distance*** is defined on any two parametrized curves $P, Q : [0,1] \to \mathbb{R}^d$ as $\inf_{f,g} \max_{t\in[0,1]} \|P(f(t)) - Q(g(t))\|$, where f and g—refered to as ***traversals***—range over all non-decreasing surjective functions from $[0,1]$ to $[0,1]$. We denote by $P[s,t]$ the ***subcurve*** of P starting at $P(s)$ and ending at $P(t)$ for $0 \leq s \leq t \leq 1$. A curve C that has Fréchet distance Δ to some such $P[s,t]$ covers every point $p \in [s,t]$. For any curve C in $\mathbb{R}^d$ we define

$$\mathrm{Cov}_P(C,\Delta) = \left(\bigcup_{0\leq s\leq t\leq 1,\, \mathrm{d}_{\mathcal{F}}(P[s,t],C)\leq\Delta} [s,t] \right) \subset [0,1]$$

as the Δ-***coverage*** of C (refer to Figure 2). This subtrajectory clustering problem is naturally interpreted as a set cover problem, where $[0,1]$ is the ground set and the set family consists of the Δ-coverage $\mathrm{Cov}_P(C,\Delta)$ for any curve C in $\mathbb{R}^d$ of complexity l.

Brüning et al. [2] showed that under the special assumption $l = 2$ any optimum solution $\mathcal{C}$ of size k to this set cover problem for given polygonal curve P and $\Delta > 0$ can be transformed to a set of curves which each start and end in so called extremal points of the 8Δ-free space of a simplification S of P with itself. The Δ-***free space*** of two curves P and Q is defined as the sublevel-set of pointwise distances less than Δ, i.e., $D_\Delta(P,Q) = \{(x,y) \mid \|P(x) - Q(y)\| \leq \Delta\}$. Monotone paths (in x and y) in the Δ-free space correspond to subcurves of P and Q with Fréchet distance at most Δ. ***Extremal points*** are local maxima/minima in the cells of the Δ-free space. This then results in a bicriteria approximation algorithm, with approximation guarantees of 11Δ in the distance threshold and $O(kl\log(kl))$ in the solution size for any l, as any center in a solution with $l > 2$ can be broken up into its edges resulting in a solution at the expense of an additional factor l in its cardinality.

3 Our contribution

The split of center curves into single edges as suggested by Brüning et al. [2] is not particularly desirable in practice. The focus of this abstract is the extension to curves of non-constant complexity $l \in \mathbb{N}_{\geq 2}$. For this we generate a set of candidate center curves based on a simplification of the input curve that are defined by extremal points. The size of the candidate set is in $O(kn^3 l)$. Details are diverted to the full version. Subsequent to the first publication of our results on arxiv van der Hoog et al. [17] observed that it suffices to consider candidates defined by pairs (a, b) of extremal coordinates such that 2^i other extremal coordinates lie in $[a, b]$ for some $i \in \mathbb{N}$. This reduces the cardinality of the candidate set to $O(n^2 \log n)$ transforming each center curve in any solution into at most two new center curves of similar complexity. This candidate set defines the set cover instance which is then solved via known techniques resulting in a bicriteria approximation algorithm, with approximation guarantees of 11Δ in the distance threshold and $O(k \log(n))$ in the solution size. The algorithm has a total running time of $O(kn^3 \log^3 n)$.

Implementation Details Our `C++`-implementation can be found at [6]. The algorithm is given a set $\{P_1, \ldots, P_m\}$ of curves and three parameters Δ_{simp}, Δ_{free} and l. It first computes simplifications S_i of P_i for all i with parameter Δ_{simp}, and then computes the Δ_{free}-free space of S_i and S_j as well as their extremal points for all pairs $(i, j) \in \{1, \ldots, m\}^2$. Next it computes all pairs of extremal coordinates that (i) lie on the same curve, (ii) the resulting subcurve has complexity at most l and (iii) there is a power of 2 of other extremal coordinates in the interval between them. For every such pair of extremal coordinates (a, b) on curve S_i we compute the Δ_{free}-coverage of $S_i[a, b]$ via the computed Δ_{free}-spaces of S_i with any other S_j. This Δ_{free}-coverage is a subset of $\{1, \ldots, m\} \times [0, 1]$ and defines the set cover instance. This set cover instance we solve via a greedy set cover algorithm that iteratively picks and adds the candidate which maximizes the arc-length (computed on each S_i) of the additional coverage. This step is repeated until $\{1, \ldots, m\} \times [0, 1]$ is covered.

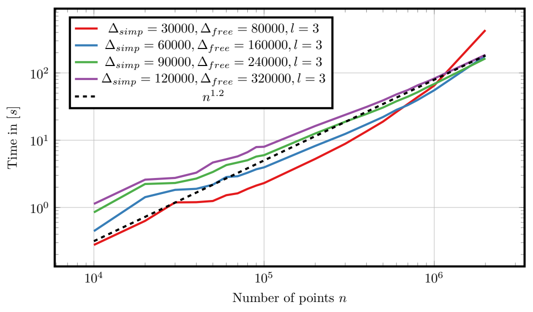
We recursively pick the candidate maximizing the arc-length—instead of the number of intervals of the induced arrangement of coverages—of the added Δ-coverage. This follows the Δ-coverage maximization discussion in [2] and allows us to stop the greedy algorithm after a small number of rounds and still have a partial solution that covers a large fraction of the input. Further, we introduced the two parameters Δ_{simp} and Δ_{free} to test the stability of the threshold parameter Δ in both the simplification and free-space computation step. Given some Δ, setting $\Delta_{simp} = 3\Delta$ and $\Delta_{free} = 8\Delta$ reflects the theoretical results.

4 Experiments

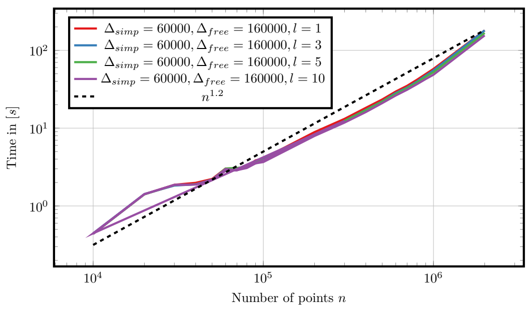
All experiments were conducted on a Linux system with 16GB of memory with an Intel `i5-9600` CPU, a decent CPU with 6 cores, but far from the fastest hardware available. All code was compiled with `clang` version `11.1` with `-O3` optimization flags enabled.

4.1 Ocean Drifters

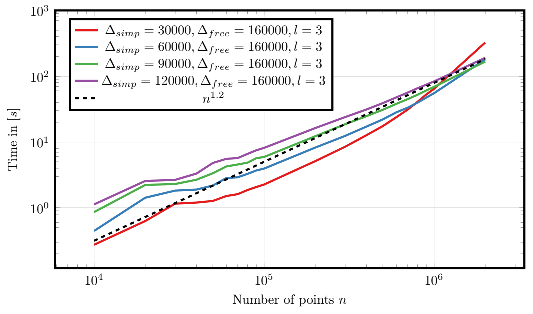
We apply our algorithm to trajectories from the NOAA Global Drifter Program [13]. This is a comprehensive data set consisting of almost 20 000 ocean surface drifters that have been released across the ocean as far back as 1979. For the evaluation, we focus on the subset of trajectories consisting of all drifters recorded in the last years (2022–2024). This data set consists of 5500 different trajectories which consist on average of 500 points resulting in



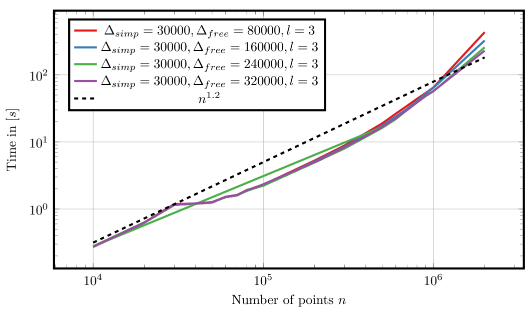
(a) varying Δ_{simp} and Δ_{free} with $3:8$ ratio.



(b) varying l.



(c) varying Δ_{simp}.



(d) varying Δ_{free}.

Figure 3 Influence of different combinations of parameters on the running time evaluated on the data set from the NOAA Global Drifter Program [13].

a total input complexity of $n \geq 2 \cdot 10^6$. Refer to Figure 1 in which the data set and the computed clustering with $\Delta_{simp} = 20km$, $\Delta_{free} = 400km$ and $l = 20$ is depicted.

Evaluation We apply our techniques with a range of radii with Δ_{simp} between $5km$ and $120km$, Δ_{free} between $10km$ and $320km$, as well as a range of complexity bounds with l between 1 and 10 on differently sized subsets of the input data. Figure 3 shows the running times. We observe that the running time appears to be mostly independent of the exact values of Δ_{simp} and Δ_{free}, and scales favorably in n compared to the theoretical results. In the $n \leq 10^5$ regime, it appears to scale near-linear with an observed running time of roughly $O(n^{1.2})$. With increasing n it approaches roughly quadratic complexity ($O(n^2)$) compared to the theoretical running time of $O(kn^3 \log^3 n)$. In addition, we empirically evaluate the approximation ratio of our set cover algorithm using the size of a greedily computed independent set as a lower bound. We observe that for all tested instances the approximation ratio is less than 3.

4.2 Full-Body Motion Tracking Data

Motion Segmentation finds applications in many different fields such as robotics, sports analysis or traffic monitoring [14]. We apply our techniques to this problem on the CMU data set [15]. This data set consists of motion tracking data of 31 different joint-trackers on different subjects doing sports (trials) ranging through different activities (refer to Figure 4). We interpret these as trajectories in 93-dimensional Euclidean space by concatenating the three-dimensional coordinates of all joints back to back to form a pose. Each trial consists of up to 10^4 poses. We then apply our subtrajectory clustering algorithm with $\Delta_{simp} = 0.8$ and

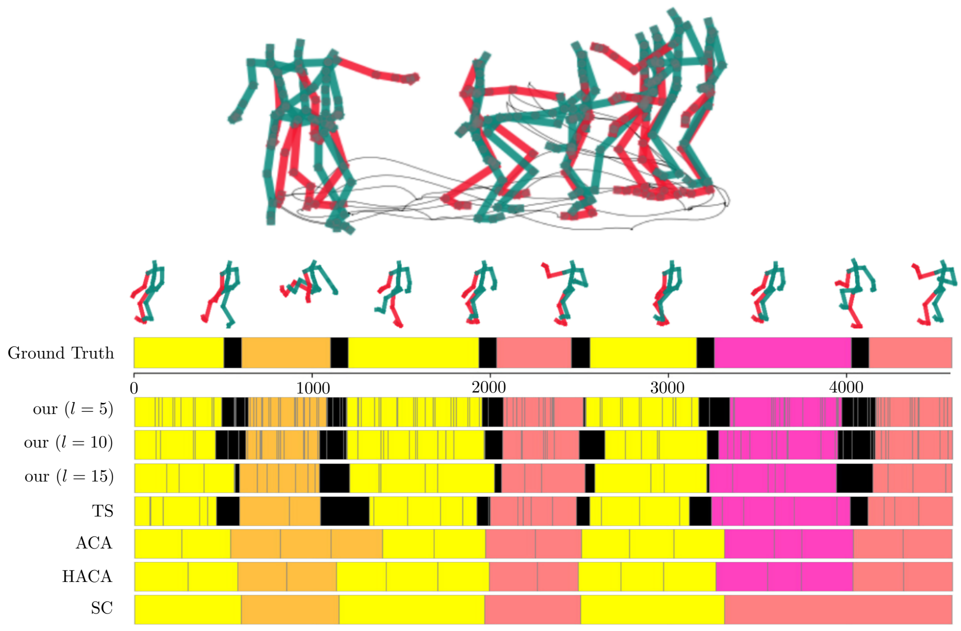


Figure 4 Trial `01` of subject `86` with its ground truth labels. Colors correspond to the labels `walk` (yellow), `jump` (orange), `punch` (light red), `kick` (magenta) and `transition` (black). Beneath the resulting labeling using our techniques with different values for l, temporal segmentation (TS), (hierarchical) aligned cluster algorithm (HACA/ACA) and spectral clustering (SC) [9, 22, 21], with gray lines corresponding to the start/end of frequent patterns.

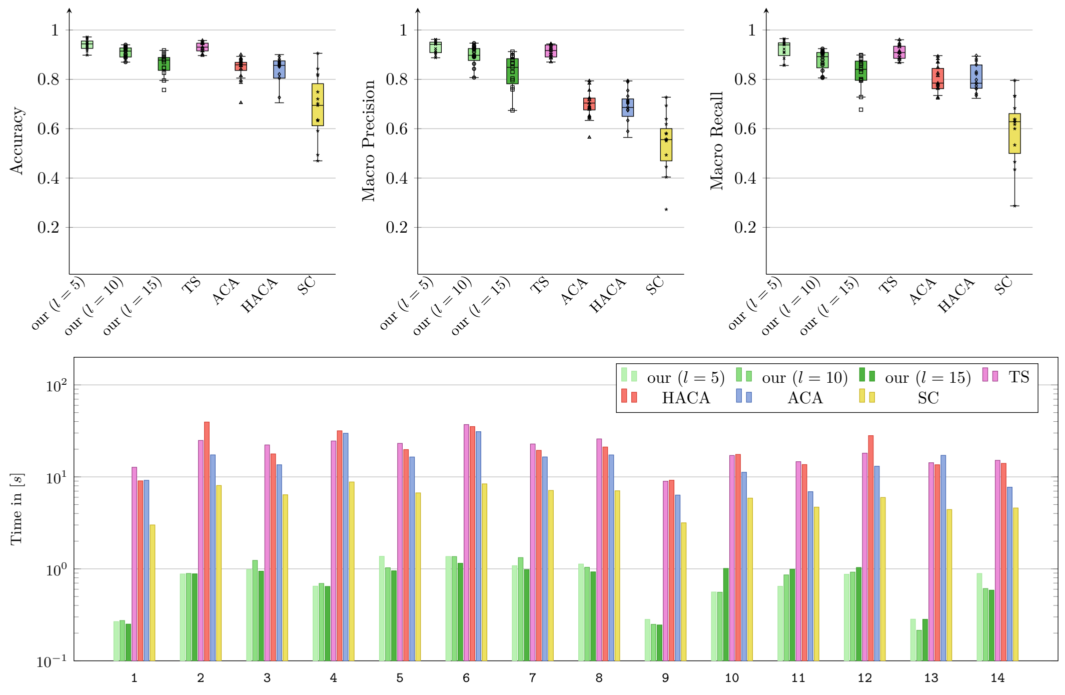


Figure 5 Quantative analysis on trial `1` to `14` of subject `86` from [15] and comparison of our techniques to temporal segmentation (TS), (hierarchical) aligned cluster algorithm (HACA/ACA) and spectral clustering (SC) from [9, 22, 21].

$\Delta_{free} \approx 1.35$ and a complexity bound l between 5 and 15, where the exact parameters have been identified via an exhaustive search to yield the best accuracy for the given complexity bound l. The output consists of a set of curves that act as cluster centers. For each of these centers we identify the ground truth label that best corresponds to this center and label all points in its Δ_{free}-Coverage with the identified label. Whenever different labels are assigned to a point along the curve we mark it as a transition between motions.

Evaluation The resulting labeling can be seen in Figure 4. Observe in particular that an increase in l decreases the number of patterns identified with the total number of labeled segments approaching that of the state-of-the-art, while the accuracy decreases only slightly. We compute the accuracy of the resulting segmentations on ground truth data from [9] and compare this accuracy with the accuracy of the temporal segmentation approach (TS) discussed in [9] as well as the aligned cluster algorithm (ACA), hierarchical aligned cluster algorithm (HACA) and spectral clustering (SC) discussed in [22, 21]. The resulting accuracies can be seen in Figure 5. The quantitative accuracy of our techniques compares well to the state-of-the-art techniques, with a (roughly) tenfold improvement in the running time.

5 Discussion

We observe that in practice the algorithm is much faster and yields better solutions than what could be expected from the theory. We partially attribute this to the fact that unlike in the worst case analysis the number of non-empty cells in the free-space is less than n^2. This reduces the number of extremal points and the complexity of the computed coverage. This suggests that analyses with additional input assumptions such as c-packedness [8] or λ-low-density [7] could result in a theoretically founded explanation of the observed running time. We further observe that the extension to non-constant complexity center curves indeed allows the algorithm to capture more interesting behaviour compared to when the center curves are restricted to constant/lower-complexity center curves.

This provides evidence that the problem formulation by [3] is practically viable and serves as a versatile tool for analyzing large amounts of spatio-temporal data.

References

1 Pankaj K. Agarwal, Kyle Fox, Kamesh Munagala, Abhinandan Nath, Jiangwei Pan, and Erin Taylor. Subtrajectory clustering: Models and algorithms. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '18, page 75–87, 2018. doi:10.1145/3196959.3196972.

2 Frederik Brüning, Jacobus Conradi, and Anne Driemel. Faster Approximate Covering of Subcurves Under the Fréchet Distance. In *30th Annual European Symposium on Algorithms (ESA 2022)*, volume 244 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:16, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2022.28.

3 Frederik Brüning, Hugo Akitaya, Erin Chambers, and Anne Driemel. Subtrajectory clustering: Finding set covers for set systems of subcurves. *Computing in Geometry and Topology*, 2(1):1:1–1:48, Feb. 2023. doi:10.57717/cgt.v2i1.7.

4 Maike Buchin, Bernhard Kilgus, and Andrea Kölzsch. Group diagrams for representing trajectories. *International Journal of Geographical Information Science*, 34(12):2401–2433, 2020. doi:10.1080/13658816.2019.1684498.

5 Maike Buchin and Carola Wenk. Inferring movement patterns from geometric similarity. *J. Spatial Inf. Sci.*, 21(1):63–69, 2020. doi:10.5311/JOSIS.2020.21.724.

6 Jacobus Conradi and Anne Driemel. Github repository. https://github.com/JacobusTheSecond/clustering, 2025.

7 Mark de Berg, A. Frank van der Stappen, Jules Vleugels, and Matthew J. Katz. Realistic input models for geometric algorithms. *Algorithmica*, 34(1):81–97, 2002. doi:10.1007/S00453-002-0961-X.

8 Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the fréchet distance for realistic curves in near linear time. *Discret. Comput. Geom.*, 48(1):94–127, 2012. doi:10.1007/S00454-012-9402-Z.

9 Björn Krüger, Anna Vögele, Tobias Willig, Angela Yao, Reinhard Klein, and Andreas Weber. Efficient unsupervised temporal segmentation of motion data. *IEEE Transactions on Multimedia*, 19(4):797–812, 2017. doi:10.1109/TMM.2016.2635030.

10 Vojtěch Kubelka, Brett K Sandercock, Tamás Székely, and Robert P Freckleton. Animal migration to northern latitudes: environmental changes and increasing threats. *Trends in ecology & evolution*, 37(1):30–41, 2022.

11 Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*, pages 593–604, 2007. doi:10.1145/1247480.1247546.

12 Anqi Liang, Bin Yao, Bo Wang, Yinpei Liu, Zhida Chen, Jiong Xie, and Feifei Li. Sub-trajectory clustering with deep reinforcement learning. *The VLDB Journal*, pages 1–18, 2024.

13 Rick Lumpkin and Luca Centurioni. Global drifter program quality-controlled 6-hour interpolated data from ocean surface drifting buoys. NOAA National Centers for Environmental Information, 2019. doi:10.25921/7ntx-z961.

14 Jana Mattheus, Hans Grobler, and Adnan M. Abu-Mahfouz. A review of motion segmentation: Approaches and major challenges. In *2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, pages 1–8, 2020. doi:10.1109/IMITEC50163.2020.9334076.

15 C MoCap. Carnegie mellon university graphics lab motion capture database, 2007. URL: http://mocap.cs.cmu.edu/.

16 Tammy Tabe. Climate change migration and displacement: Learning from past relocations in the pacific. *Social Sciences*, 8(7):218, 2019.

17 Ivor van der Hoog, Thijs van der Horst, and Tim Ophelders. Faster and deterministic subtrajectory clustering, 2024. arXiv:2402.13117.

18 Sheng Wang, Zhifeng Bao, J Shane Culpepper, and Gao Cong. A survey on trajectory data management, analytics, and learning. *ACM Computing Surveys (CSUR)*, 54(2):1–36, 2021. doi:10.1145/3440207.

19 Laura J Wilson, Christopher J Fulton, Andrew McC Hogg, Karen E Joyce, Ben TM Radford, and Ceridwen I Fraser. Climate-driven changes to ocean circulation and their inferred impacts on marine dispersal patterns. *Global ecology and biogeography*, 25(8):923–939, 2016.

20 Guan Yuan, Penghui Sun, Jie Zhao, Daxing Li, and Canwei Wang. A review of moving object trajectory clustering algorithms. *Artificial Intelligence Review*, 47(1):123–144, 2017. doi:10.1007/s10462-016-9477-7.

21 Feng Zhou, Fernando De la Torre, and Jessica K. Hodgins. Aligned cluster analysis for temporal segmentation of human motion. In *2008 8th IEEE International Conference on Automatic Face & Gesture Recognition*, pages 1–7, 2008. doi:10.1109/AFGR.2008.4813468.

22 Feng Zhou, Fernando De la Torre, and Jessica K Hodgins. Hierarchical aligned cluster analysis for temporal clustering of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(3):582–596, 2012.

Partial Level Planarity Parameterized by the Size of the Missing Graph[*]

Thomas Depian[1], Simon D. Fink[1], Boris Klemz[2], Robert Ganian[1], Martin Nöllenburg[1], and Marie Diana Sieper[2]

1 Algorithms and Complexity Group, TU Wien, Austria
`{tdepian, sfink, rganian, noellenburg}@ac.tuwien.ac.at`

2 Algorithms and Complexity Group, Universität Würzburg, Germany
`firstname.lastname@uni-wuerzburg.de`

Abstract

A level planar drawing of a graph G is crossing-free, has y-monotone edges, and uses levels specified in the input as y-coordinates of vertices. In Partial Level Planarity (PLP), we are given a level planar drawing of a subgraph H of G and are asked to extend it to a drawing of G. Results by Brückner and Rutter [SODA 2017] and Klemz and Rote [ACM Trans. Alg. 2019] show that PLP remains NP-complete even for severely restricted instances, which makes it resistant against parameterized tractability for most well-known parameters. In this paper, we use the size of the missing graph, i.e., of $G \setminus H$, as a parameter to identify fixed-parameter tractable fragments of PLP.

1 Introduction

For any type of graph representation, in its partial representation extension problem, we are given a graph G together with a representation Γ_H of a subgraph $H \subseteq G$. We seek a representation Γ_G of G that *extends* Γ_H, i.e., when restricted to H, coincides with Γ_H. Depending on the representation, this problem presents a rich diversity in terms of computational complexity. For planar straight-line drawings, the problem is NP-hard [30] and even $\exists\mathbb{R}$-complete [29], while being linear-time solvable for planar topological or orthogonal drawings [1–3, 17]. For intersection representations of, e.g., interval graphs [24–26], permutation graphs [23], and circle graphs [10], the problem is efficiently solvable, while it is NP-hard, e.g., for contact representations of geometric objects [9]. In the NP-hard case, a more refined analysis in terms of fixed parameter tractability often yields interesting insights [12, 13, 16, 19, 20]; a natural approach employed in this setting is to parameterize by the size of the missing subgraph.

In this paper, we consider the extension problem for Level Planarity, often called Partial Level Planarity (PLP). In Level Planarity, we seek a planar topological drawing of a graph in which each vertex has a prescribed y-coordinate, also called its *level*, and all edges are y-monotone; see Figure 1b. While Level Planarity can be tested in linear time [21, 22], its extension variant PLP is NP-complete even for restricted instances: Brückner and Rutter [8] showed NP-hardness for subdivisions of triconnected planar graphs with bounded degree and Klemz and Rote [27] for disjoint unions of constant-length paths (see [28]). The latter rules out an efficient parametrization of PLP with respect to most classical structural parameters. Blazej, Klemz, Klesen, Sieper, Wolff, and Zink [6] considered the number of levels as a parameter for generalizations and specializations of PLP. Here, we

[*] TD, SF, RG, and MN acknowledge support from the Vienna Science and Technology Fund (WWTF) [10.47379/ICT22029]; RG also acknowledges support from the Austrian Science Fund (FWF) [10.55776/Y1329].

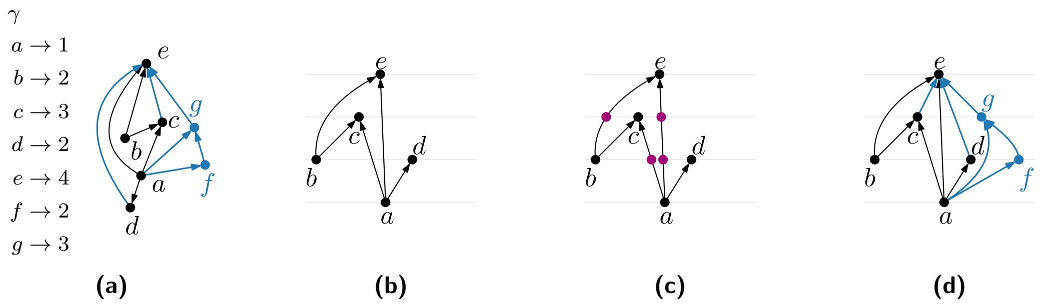


Figure 1 **(a)** Level graph $\mathcal{G}$ with a subgraph $\mathcal{H} \subseteq \mathcal{G}$ (black) and **(b)** a level planar drawing $\Gamma_\mathcal{H}$. **(c)** By introducing the lilac subdivision vertices, we make $\mathcal{H}$ proper. The drawing $\Gamma_\mathcal{H}$ from **(b)** is extended in **(d)** to a drawing $\Gamma_\mathcal{G}$ of $\mathcal{G}$ by the new vertices and edges marked in blue.

will investigate the complexity of PLP parameterized by the size of the missing (also called *new*) subgraph. As our first result, in Section 3, we show that PLP is FPT[1] parameterized by the number of missing vertices when every edge connects adjacent levels. We also lift the latter restriction at the cost of using the number of missing edges as a more restrictive parameter. Finally, in Section 4, we relax our parametrization to be close to the number of missing vertices again, but require the instance to be biconnected. This can be seen as a first step towards a general fixed-parameter algorithm that only parametrizes by the number of missing vertices (work in progress), and is in line with previous solutions that approach problems at decreasing levels of connectivity.

Due to space constraints, technical details are deferred to an upcoming full version.

2 Preliminaries

Let G be a simple, directed, acyclic, and planar graph with vertex set $V(G)$ and edge set $E(G)$. We use V and E if G is clear from the context. Let $\gamma\colon V \to [\ell] = \{1, 2, \ldots, \ell\}$ be a *level assignment* of G, i.e., a function that assigns each vertex $v \in V$ to one of $\ell \leq |V|$ *levels* with $\gamma(u) < \gamma(v)$ for all $uv \in E$. The vertices on level $i \in [\ell]$ are denoted by $V_i \coloneqq \{v \in V \mid \gamma(v) = i\}$. We call an edge $uv \in E$ *short* if $\gamma(u) + 1 = \gamma(v)$, or *long* otherwise and we say it *spans* the levels between $\gamma(u)$ and $\gamma(v)$ (excluding $\gamma(u)$ and $\gamma(v)$). A *proper* level graph has only short edges; see Figure 1c. For a *level graph* $\mathcal{G} = (G, \gamma)$, a *level planar drawing* is a planar drawing $\Gamma_\mathcal{G}$ of G with $y(\Gamma_\mathcal{G}(v)) = \gamma(v)$ for each $v \in V$ and where each edge is a y-monotone curve. A *level planar embedding* is the equivalence class of level planar drawings where each horizontal line corresponding to a level is intersected by vertices on and edges spanning the level in the same order. We use $\prec_i$ to denote the total order of the vertices on level $i \in [\ell]$, also dropping the index if i is clear from the context. A level graph $\mathcal{H} = (H, \gamma_H)$ is a *subgraph* of $\mathcal{G} = (G, \gamma_G)$, denoted as $\mathcal{H} \subseteq \mathcal{G}$, if $H \subseteq G$ and $\gamma_H = \gamma_G|_H$. Proper PLP and Bicon PLP denote the restricted variants of PLP where $\mathcal{G}$ is proper or G and H are biconnected, respectively. For an instance $\mathcal{I} = (\mathcal{G}, \mathcal{H} \subseteq \mathcal{G}, \Gamma_\mathcal{H})$ of PLP, we let $n \coloneqq |V(G)|$ denote the number of vertices in G and remark that $|E(G)|$ is linear in n but $|\mathcal{I}|$ can be quadratic in n for non-proper level graphs. We refer to the vertices and edges of H as *old*, and all other vertices and edges of G as *new* or *missing*. Furthermore, we indicate with V_{add} and E_{add} the set of new vertices and edges, and with n_{add}

[1] We assume familiarity with standard notions from graph theory [14] and parameterized complexity [11].

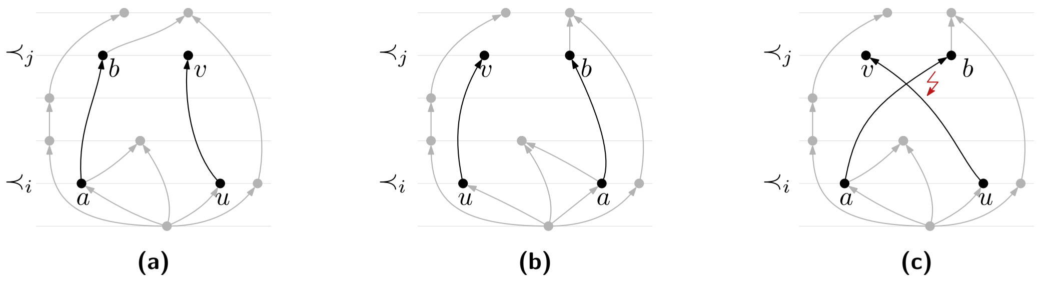


Figure 2 The total orders $\prec_i$ and $\prec_j$ imply planarity in **(a)** and **(b)**, but the crossing from **(c)**.

and m_{add} their size, respectively. We let E^H_{add} denote the set of new edges incident to two old vertices, i.e., $E^H_{\text{add}} := \{uv \in E_{\text{add}} \mid u, v \in V(H)\}$. To measure how "incomplete" the provided partial solution is, we consider the following natural parameters that have also been used before in other extension problems [5, 12, 15, 16, 19]: the *vertex+edge deletion distance* $\kappa := n_{\text{add}} + |E^H_{\text{add}}|$, and the *numbers of missing vertices* n_{add} or *edges* m_{add}.

3 PLP Parameterized by the Number of New Vertices or Edges

We now present an FPT-algorithm for PROPER PLP parameterized by n_{add}. We first handle the new edges with two old endpoints E^H_{add}. As $\mathcal{G}$ is proper, the drawing $\Gamma_{\mathcal{H}}$ defines a unique embedding for $G[V(H)] = H \cup E^H_{\text{add}}$ in the form of per-level vertex orders for $V(H)$. Randerath et al. [31] give the following necessary condition for planarity of such embeddings; see also Figure 2.

▶ **Observation 3.1.** *In any level planar embedding of a level graph $\mathcal{G} = (G, \gamma)$ with $uv, ab \in E$, $u \neq a$, $v \neq b$, $\gamma(u) = \gamma(a) = i$, and $\gamma(v) = \gamma(b) = j$, we have $u \prec_i a \Leftrightarrow v \prec_j b$.*

This criterion is also sufficient for proper level graphs [7, 31], enabling us to efficiently test whether the unique embedding of $G[V(H)] = H \cup E^H_{\text{add}}$ induced by $\Gamma_{\mathcal{H}}$ is planar. This allows us to consider these edges for the remainder of the algorithm as old.

We continue with guessing an order $\prec_i^{V_{\text{add}}}$ among the new vertices for each level $i \in [\ell]$ using $\prod_{i=1}^{\ell} |V_i \cap V_{\text{add}}|! \leq n_{\text{add}}!$ branches. In each branch and on each level $i \in [\ell]$, $\prec_i^H$ and $\prec_i^{V_{\text{add}}}$ fix an order among the old and new vertices, respectively. It remains to merge them into a single order that describes a level planar embedding. For this, we use a 2SAT formula that builds on Observation 3.1. To that end, to formulate level planarity as a Boolean satisfiability problem, we use a variable $x_{u,a}$ $(= \neg x_{a,u})$ per ordered pair of vertices u, a on the same level, which represents whether $u \prec a$. To obtain a total vertex order for each level, we also have to ensure transitivity, that is $x_{a,b} \wedge x_{b,c} \Rightarrow x_{a,c}$ for all $a, b, c \in V_i, i \in [\ell]$. Note that while this additional condition cannot be represented in a 2SAT formula, it has been shown that it can be elided if the 2SAT formula solely consists of the conditions derived from Observation 3.1 [7, 18, 31]. As our partially fixed vertex orders do yield further conditions (they actually fix some variable values), we need a different way of ensuring transitivity while still obtaining a polynomial-time solvable formula.

For any triple $a, b, c \in V_i$ with $i \in [\ell]$, for which we have to ensure a transitive order, at least two vertices are of the same type (new or old). As the orders among old and new vertices are determined by the partial drawing and our branching, respectively, at least one variable of $x_{a,b} \wedge x_{b,c} \Rightarrow x_{a,c}$ is predetermined by either $\prec_i^H$ or $\prec_i^{V_{\text{add}}}$. This allows us to represent the transitivity condition using 2SAT as shown by the following lemma, where the

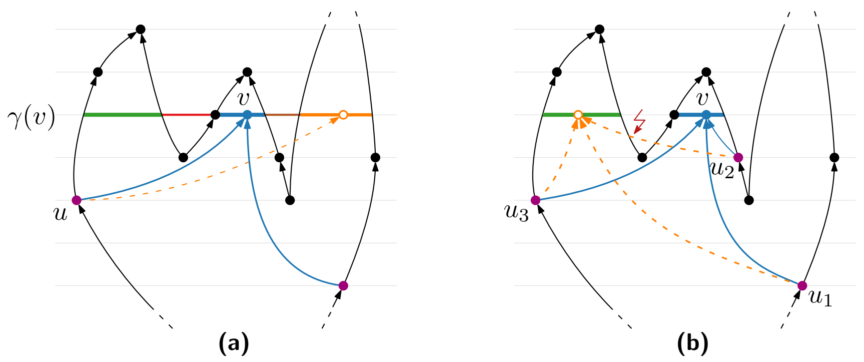


Figure 3 **(a)** Intervals of the new vertex v. Only if placed in the green or blue interval, v is visible from all lilac adjacent old vertices. **(b)** If we remove the new edge u_2v and retain only u_1v and u_3v, v could also be placed in the green interval, making re-inserting u_2v impossible.

running time follows from the fact that satisfiability for an instance of 2SAT can be checked in time linear in the number of variables [4].

▶ **Lemma 3.2.** *Let $\mathcal{G} = (G, \gamma)$ be a proper level graph with its vertices partitioned into two sets $A \cup B = V$ and let $\prec_i^A$ and $\prec_i^B$ be total orders of $V_i \cap A$ and $V_i \cap B$, respectively, for all $i \in [\ell]$. In $\mathcal{O}(n^2)$ time, we can find a level planar embedding of $\mathcal{G}$ where $\prec_i$ extends $\prec_i^A$ as well as $\prec_i^B$ for all $i \in [\ell]$, or report that no such embedding exists.*

Combining this with our $\mathcal{O}(n_{\text{add}}!)$ branches, we obtain the following.

▶ **Theorem 3.3.** PROPER PLP *can be solved in time $\mathcal{O}(n_{add}! \cdot n^2)$ and is thereby fixed-parameter tractable when parameterized by the number of missing vertices n_{add}.*

To generalize this approach to non-proper instances, we first guess a total order $\prec^{E_{\text{add}}}$ of all new edges E_{add} using $\mathcal{O}(m_{\text{add}}!)$ branches. Then, we subdivide all long edges to obtain a proper level graph $\mathcal{G}'$ with $\mathcal{O}(n^2)$ vertices and edges. Observe that the order $\prec^{E_{\text{add}}}$ on new edges (and similarly $\prec^H$ on the old graph) now induces per-level orders $\prec_i^{V_{\text{add}}}$ and $\prec_i^H$ for all new and old vertices, respectively, including the ones created through the subdivision process. Using these orders, we can now apply the algorithm from Lemma 3.2 on $\mathcal{G}'$, yielding the following theorem.

▶ **Theorem 3.4.** PLP *can be solved in time $\mathcal{O}(m_{add}! \cdot n^4)$ and is thereby fixed-parameter tractable when parameterized by the number of missing edges m_{add}.*

4 BICON PLP Parameterized by the Vertex+Edge Deletion Distance

We now show that PLP is FPT when G and H are biconnected using $\kappa = n_{\text{add}} + |E_{\text{add}}^H|$ as parameter. Note that the number of new edges connecting the new graph to the old one is not bounded by a function in κ, which is the main obstacle to applying Theorem 3.4. To bound this number, we will first show that we can guess, for each connected component of $G[V_{\text{add}}]$, an assignment to a $\Gamma_{\mathcal{H}}$-face into which it shall be drawn. Then, we show that, when walking around the boundary of a $\Gamma_{\mathcal{H}}$-face, the adjacent new vertices cannot alternate too much. We show for the remaining consecutive new edges with the same new endpoint that we only need to retain the first and last edge, allowing us to bound their number by κ. In our algorithm, we need to prevent new vertices v from being placed inside certain *intervals*, i.e., between two given old vertices $u_1 \prec^{H'} u_2$ on the same level in the proper level graph $\mathcal{H}'$

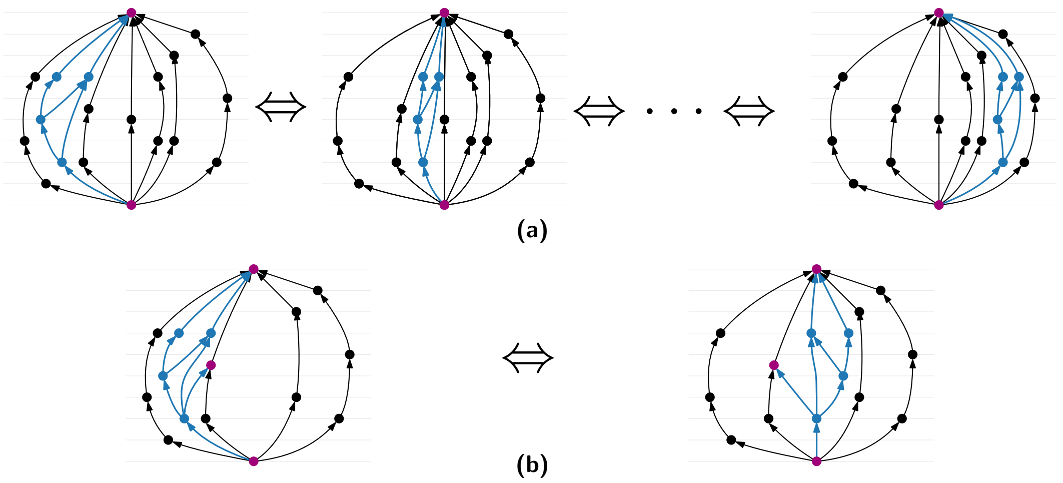


Figure 4 A missing component C (blue) and $N_H(C)$ (lilac): **(a)** $|N_H(C)| = 2$ and **(b)** $|N_H(C)| \geq 3$.

obtained by subdividing long edges in $\mathcal{H}$; see also Figure 3a. Note that this is equivalent to forcing v to be in one of the remaining intervals. In the 2Sat formulation of Theorem 3.4, this can be achieved by adding the clause $(u_1 \prec v) \Leftrightarrow (u_2 \prec v)$ for each such $u_1 \prec^{H'} u_2$. Figure 3b illustrates why this restriction is crucial to maintain correctness.

Now consider a connected component C of $G[V_{\text{add}}]$ and let $N_H(C) \subseteq V(H)$ be the set of old vertices that are neighbors of C in G. As G is biconnected, we can assume $N_H(C)$ to contain at least two vertices. If $|N_H(C)| = 2$, the number of edges between C and H is bounded by κ, while the vertices in $N_H(C)$ can be incident to $\Theta(n)$ faces of $\Gamma_{\mathcal{H}}$; see Figure 4a. For each such face f, we can use the 2Sat formula of Theorem 3.4 to check whether C can be drawn within f. Assuming $\mathcal{G}$ to be level planar, we further have $|E(C)| = \mathcal{O}(\kappa)$. Thus, we can determine the faces $\mathcal{F}(C)$ of $\Gamma_{\mathcal{H}}$ into which C can be drawn level-planarly in $\mathcal{O}(\kappa! \cdot n^5)$ time. If $|\mathcal{F}(C)| \geq \kappa$, we can safely remove C from $\mathcal{I}$, as every solution for the resulting graph has a face into which we can draw C. Otherwise, or if $|N_H(C)| \geq 3$ (see Figure 4b), the number of faces containing C is bounded by κ, allowing us to guess the face f into which C shall be drawn in a solution $\Gamma_{\mathcal{G}}$. We can thus from now on assume that each new connected component $C \in G[V_{\text{add}}]$ is equipped with a face $\chi(C)$ of $\Gamma_{\mathcal{H}}$ it shall be embedded in, which we can ensure by adapting the 2Sat formula of Theorem 3.4 as described above. To now also bound the overall number of new edges by a function in κ, we show that the following properties hold for any positive instance with predetermined faces.

(i) Two new vertices in the same $\Gamma_{\mathcal{H}}$-face can share at most two old neighbors on the boundary of the face; see Figure 5a. As $\left|E^H_{\text{add}}\right| \leq \kappa$, each $\Gamma_{\mathcal{H}}$-face thus has a bounded number of old vertices adjacent to more than one new vertex or incident to a new edge of E^H_{add}. Removing these vertices from the closed walk that goes around the face boundary thus splits it into *segments*, see Figure 5b, whose number is bounded by $\mathcal{O}(\kappa^2)$.

(ii) We *label* the old vertices $w \in V(H)$ on each segment with tuples (v, d) that capture their adjacent new vertices $v \in V_{\text{add}}$ and whether they lie above $(d = \uparrow)$ or below $(d = \downarrow)$ the vertex w. Along a segment, any two labels cannot alternate, that is, they can neither form a u-v-u-v-sequence as in Figure 6a nor a $\uparrow$-$\downarrow$-$\uparrow$-$\downarrow$-sequence as in Figure 6b. Thus, the number of switches between labels along a segment is bounded by a function in κ, but we

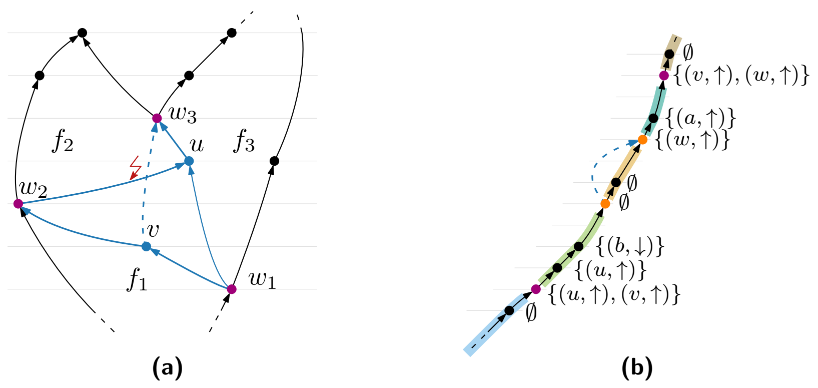


Figure 5 **(a)** The new edges incident to u create three faces f_i, making a planar connection for v impossible. **(b)** Part of a face f split into segments visualized by color. Vertices are shown with their labels and those incident to two new edges or to an edge of E_{add}^{H} are lilac and orange, respectively.

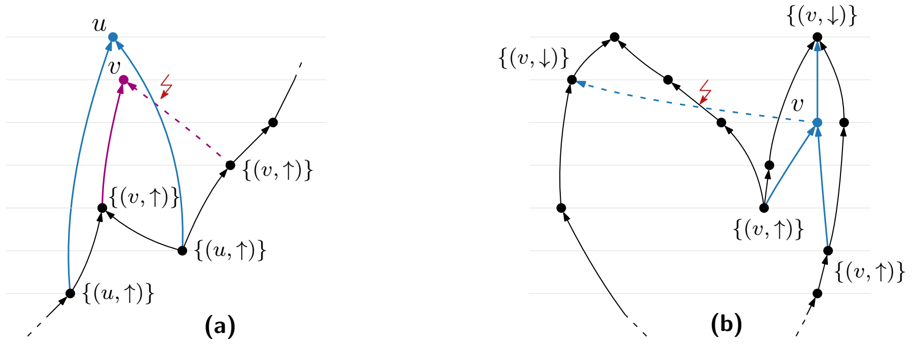


Figure 6 Alternating the **(a)** vertex or the **(b)** level information of a label implies a crossing.

may still have arbitrary long streaks of the same label.

(iii) For three consecutive identical labels on the vertices u_1, u_2, and u_3 along the same segment we now aim to delete the edge $u_2v \in E_{\text{add}}$ (or vu_2) causing the middle label. For this operation to be safe, i.e., to guarantee that we can later re-insert the edge u_2v, we must ensure that v is embedded in an interval that is visible from u_2 to prevent a situation as in Figure 3b. A *visible* interval for some old vertex u in the face f consists of two consecutive old vertices $w_1 \prec w_2$ on another level where both the edges vw_1 and vw_2 (or their reversal) can be added level-planarly to $\Gamma_{\mathcal{H}'}$, where $\mathcal{H}'$ denotes the subdivided $\mathcal{H}$. We can find these intervals in a preprocessing step, and the constraint of v to lie within such intervals can again be encoded in the 2SAT formula. This constraint is necessary as argued above, both for the instance before and after the deletion of u_2v. In particular, observe that a hypothetical solution must place v in one of these intervals. Due to the constraint, we can assume that the interval in which v is placed is visible from u_2. To also argue that it is sufficient after the deletion, we consider Figure 7. On the one hand, Figures 7a and b underline that there cannot exist a vertex $w \neq u_2 \in V(H)$ between u_1 and u_3 that is incident to a new edge that blocks visibility for u_2, as this violates the selection of u_1, u_2, and u_3. On the other hand, Figure 7c shows that any other new edge $ab \in E_{\text{add}}$ that blocks visibility for u_2 yields a contradiction.

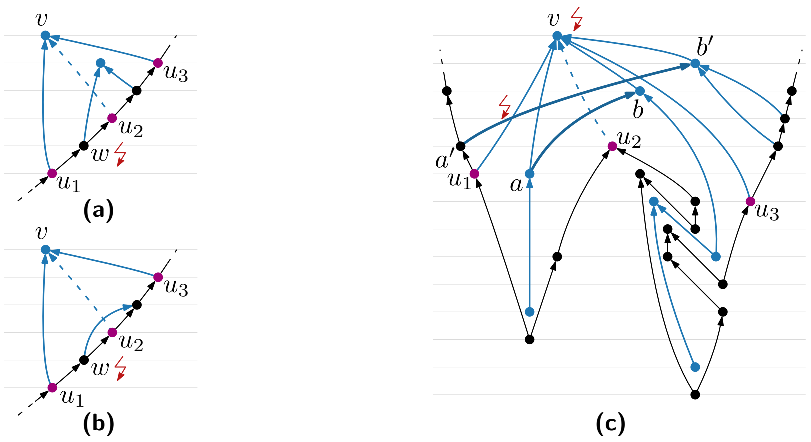


Figure 7 Deleting the edge u_2v is safe: There cannot exist a vertex $u_2 \neq w \in V(H)$ between u_1 and u_3 incident to a new edge as it has **(a)** no label by the selection of u_1, u_2, and u_3 and **(b)** cannot be incident to an edge from E^H_{add} by the definition of a segment. **(c)** Any other new edge that blocks visibility for u_2 either crosses with u_1v or u_3v, as shown with the edge $a'b'$, or is completely inside the "triangular cycle" induced by v and the path on the segment between u_1 and u_3, as shown with the edge ab. The former case implies that the instance is a no-instance, and the latter case that v is a cut-vertex in G, which contradicts the fact that G is biconnected.

Altogether, this allows us to show that deleting the middle edge is safe. This further allows us to bound the number of label switches and the remaining vertices with labels inbetween them by κ. Consequently, we can also bound the overall number of new edges in a function in κ. Thus, we can apply Theorem 3.4 to solve the instance in $\mathsf{FPT}(\kappa)$-time, also respecting the restrictions of certain new vertices to lie outside certain intervals via its 2SAT formula.

▶ **Theorem 4.1.** BICON PLP *can be solved in time* $\mathcal{O}(\kappa^\kappa \cdot \kappa \cdot (\kappa^3)! \cdot n^5)$ *and is thereby fixed-parameter tractable when parameterized by the vertex+edge deletion distance* κ*.*

References

1 Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Vít Jelínek, Jan Kratochvíl, Maurizio Patrignani, and Ignaz Rutter. Testing Planarity of Partially Embedded Graphs. In Moses Charikar, editor, *Proc. 21st ACM-SIAM Symposium on Discrete Algorithms (SODA'10)*, pages 202–221. SIAM, 2010. doi:10.1137/1.9781611973075.19.

2 Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Vít Jelínek, Jan Kratochvíl, Maurizio Patrignani, and Ignaz Rutter. Testing Planarity of Partially Embedded Graphs. *ACM Transactions on Algorithms*, 11(4):32:1–32:42, 2015. doi:10.1145/2629341.

3 Patrizio Angelini, Ignaz Rutter, and T. P. Sandhya. Extending Partial Orthogonal Drawings. *Journal of Graph Algorithms and Applications*, 25(1):581–602, 2021. doi:10.7155/jgaa.00573.

4 Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A Linear-Time Algorithm for Testing the Truth of Certain Quantified Boolean Formulas. *Information Processing Letters*, 8(3):121–123, 1979. doi:10.1016/0020-0190(79)90002-4.

5 Sujoy Bhore, Robert Ganian, Liana Khazaliya, Fabrizio Montecchiani, and Martin Nöllenburg. Extending Orthogonal Planar Graph Drawings Is Fixed-Parameter Tractable. In Erin W. Chambers and Joachim Gudmundsson, editors, *Proc. 39th International Symposium*

on Computational Geometry (SoCG'23), volume 258 of *LIPIcs*, pages 18:1–18:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.SOCG.2023.18.

6 Václav Blazej, Boris Klemz, Felix Klesen, Marie Diana Sieper, Alexander Wolff, and Johannes Zink. Constrained and Ordered Level Planarity Parameterized by the Number of Levels. In Wolfgang Mulzer and Jeff M. Phillips, editors, *Proc. 40th International Symposium on Computational Geometry (SoCG'24)*, volume 293 of *LIPIcs*, pages 20:1–20:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.SOCG.2024.20.

7 Guido Brückner, Ignaz Rutter, and Peter Stumpf. Level Planarity: Transitivity vs. Even Crossings. In Therese Biedl and Andreas Kerren, editors, *Proc. 26th International Symposium on Graph Drawing and Network Visualization (GD'18)*, volume 11282 of *Lecture Notes in Computer Science*, pages 39–52. Springer, 2018. doi:10.1007/978-3-030-04414-5_3.

8 Guido Brückner and Ignaz Rutter. Partial and Constrained Level Planarity. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, January 2017. doi:10.1137/1.9781611974782.130.

9 Steven Chaplick, Paul Dorbec, Jan Kratochvíl, Mickaël Montassier, and Juraj Stacho. Contact Representations of Planar Graphs: Extending a Partial Representation is Hard. In Dieter Kratsch and Ioan Todinca, editors, *Proc. 40th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'14)*, volume 8747 of *Lecture Notes in Computer Science*, pages 139–151. Springer, 2014. doi:10.1007/978-3-319-12340-0_12.

10 Steven Chaplick, Radoslav Fulek, and Pavel Klavík. Extending partial representations of circle graphs. *Journal of Graph Theory*, 91(4):365–394, 2019. doi:10.1002/jgt.22436.

11 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

12 Thomas Depian, Simon D. Fink, Robert Ganian, and Martin Nöllenburg. The Parameterized Complexity Of Extending Stack Layouts. In Stefan Felsner and Karsten Klein, editors, *Proc. 32nd International Symposium on Graph Drawing and Network Visualization (GD'24)*, volume 320 of *LIPIcs*, pages 12:1–12:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.GD.2024.12.

13 Thomas Depian, Simon Dominik Fink, Alexander Firbas, Robert Ganian, and Martin Nöllenburg. Pathways to Tractability for Geometric Thickness. In *Proc. 50th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'25)*, 2025. To appear. arXiv:2411.15864.

14 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate Texts in Mathematics*. Springer, 2012.

15 Eduard Eiben, Robert Ganian, Thekla Hamm, Fabian Klute, and Martin Nöllenburg. Extending Partial 1-Planar Drawings. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *Proc. 47th International Colloquium on Automata, Languages and Programming (ICALP'20)*, volume 168 of *LIPIcs*, pages 43:1–43:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.ICALP.2020.43.

16 Eduard Eiben, Robert Ganian, Thekla Hamm, Fabian Klute, and Martin Nöllenburg. Extending Nearly Complete 1-Planar Drawings in Polynomial Time. In Javier Esparza and Daniel Král', editors, *Proc. 45th Mathematical Foundations of Computer Science (MFCS'20)*, volume 170 of *LIPIcs*, pages 31:1–31:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.MFCS.2020.31.

17 Simon D. Fink, Ignaz Rutter, and T. P. Sandhya. A Simple Partially Embedded Planarity Test Based on Vertex-Addition. In *Proc. 8th SIAM Symposium on Simplicity in Algorithms (SOSA'25)*, 2025. To appear. arXiv:2410.13536.

18 Radoslav Fulek, Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Hanani-Tutte, Monotone Drawings, and Level-Planarity. In János Pach, editor, *Thirty Essays on Ge-*

ometric Graph Theory, pages 263–287. Springer, 2013. `doi:10.1007/978-1-4614-0110-0_14`.

19 Robert Ganian, Thekla Hamm, Fabian Klute, Irene Parada, and Birgit Vogtenhuber. Crossing-Optimal Extension of Simple Drawings. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *Proc. 48th International Colloquium on Automata, Languages and Programming (ICALP'21)*, volume 198 of *LIPIcs*, pages 72:1–72:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPICS.ICALP.2021.72`.

20 Thekla Hamm and Petr Hlinený. Parameterised Partially-Predrawn Crossing Number. In Xavier Goaoc and Michael Kerber, editors, *Proc. 38th International Symposium on Computational Geometry (SoCG'22)*, volume 224 of *LIPIcs*, pages 46:1–46:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPICS.SOCG.2022.46`.

21 Michael Jünger and Sebastian Leipert. Level Planar Embedding in Linear Time. In Jan Kratochvíl, editor, *Proc. 7th International Symposium on Graph Drawing and Network Visualization (GD'99)*, volume 1731 of *Lecture Notes in Computer Science*, pages 72–81. Springer, 1999. `doi:10.1007/3-540-46648-7_7`.

22 Michael Jünger and Sebastian Leipert. Level Planar Embedding in Linear Time. *Journal of Graph Algorithms and Applications*, 6(1):67–113, 2002. `doi:10.7155/JGAA.00045`.

23 Pavel Klavík, Jan Kratochvíl, Tomasz Krawczyk, and Bartosz Walczak. Extending Partial Representations of Function Graphs and Permutation Graphs. In Leah Epstein and Paolo Ferragina, editors, *Proc. 20th European Symposium on Algorithms (ESA'12)*, volume 7501 of *Lecture Notes in Computer Science*, pages 671–682. Springer, 2012. `doi:10.1007/978-3-642-33090-2_58`.

24 Pavel Klavík, Jan Kratochvíl, Yota Otachi, Ignaz Rutter, Toshiki Saitoh, Maria Saumell, and Tomás Vyskocil. Extending partial representations of proper and unit interval graphs. *Algorithmica*, 77(4):1071–1104, 2017. `doi:10.1007/s00453-016-0133-z`.

25 Pavel Klavík, Jan Kratochvíl, Yota Otachi, Toshiki Saitoh, and Tomáš Vyskočil. Extending Partial Representations of Interval Graphs. *Algorithmica*, 78(3):945–967, 2016. `doi:10.1007/s00453-016-0186-z`.

26 Pavel Klavík, Jan Kratochvíl, and Tomás Vyskocil. Extending Partial Representations of Interval Graphs. In Mitsunori Ogihara and Jun Tarui, editors, *Proc. 8th Theory and Applications of Models of Computation (TAMC'11)*, volume 6648 of *Lecture Notes in Computer Science*, pages 276–285. Springer, 2011. `doi:10.1007/978-3-642-20877-5_28`.

27 Boris Klemz and Günter Rote. Ordered Level Planarity and Its Relationship to Geodesic Planarity, Bi-Monotonicity, and Variations of Level Planarity. *ACM Transactions on Algorithms*, 15(4):1–25, 2019. `doi:10.1145/3359587`.

28 Boris Klemz and Marie Diana Sieper. Constrained Level Planarity Is FPT with Respect to the Vertex Cover Number. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *Proc. 51st International Colloquium on Automata, Languages and Programming (ICALP'24)*, volume 297 of *LIPIcs*, pages 99:1–99:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. `doi:10.4230/LIPICS.ICALP.2024.99`.

29 Anna Lubiw, Tillmann Miltzow, and Debajyoti Mondal. The Complexity of Drawing a Graph in a Polygonal Region. In Therese Biedl and Andreas Kerren, editors, *Proc. 7th International Symposium on Graph Drawing and Network Visualization (GD'18)*, volume 11282 of *Lecture Notes in Computer Science*, pages 387–401. Springer, 2018. `doi:10.1007/978-3-030-04414-5_28`.

30 Maurizio Patrignani. On Extending a Partial Straight-line Drawing. *International Journal of Foundations of Computer Science*, 17(5):1061–1070, 2006. `doi:10.1142/S0129054106004261`.

31 Bert Randerath, Ewald Speckenmeyer, Endre Boros, Peter L. Hammer, Alexander Kogan, Kazuhisa Makino, Bruno Simeone, and Ondrej Cepek. A Satisfiability Formulation of

Problems on Level Graphs. *Electronic Notes Discreete Mathematics*, 9:269–277, 2001. doi:10.1016/S1571-0653(04)00327-0.

Bounds for k-centers of point sets under L_∞-bottleneck distance

Mats Bierwirth[1], Julia Hütte[1], Patrick Schnider[1,2], and Bettina Speckmann[3]

1 Dept. of Computer Science, ETH Zürich, Switzerland
[mbierwirth | jhuette]@student.ethz.ch & patrick.schnider@inf.ethz.ch
2 Dept. of Mathematics and Computer Science, University of Basel, Switzerland
3 Dept. of Mathematics and Computer Science, TU Eindhoven, The Netherlands
b.speckmann@tue.nl

Abstract

We consider the k-center problem on the space of fixed-size point sets in the plane under the L_∞-bottleneck distance. While this problem is motivated by persistence diagrams in topological data analysis, we illustrate it as a *Restaurant Supply Problem*: given n restaurant chains of m stores each, we want to place supermarket chains, also of m stores each, such that each restaurant chain can select one supermarket chain to supply all its stores, ensuring that each store is matched to a nearby supermarket. How many supermarket chains are required to supply all restaurants? We address this questions under the constraint that any two restaurant chains are close enough under the L_∞-distance to be satisfied by a single supermarket chain. We provide both upper and lower bounds for this problem and investigate its computational complexity.

Related Version arXiv:2503.02715

1 Introduction

The *k-center problem* is a classical problem in computational geometry [2, 6, 8, 9, 16]. Given a set P of points in some metric space X, the goal is to partition them into k parts $P_1, \dots, P_k$ such that for each part P_i there is some other point $x_i \in X$ that is close to each point $p \in P_i$. While this problem has mainly been studied for points in Euclidean space, motivated by *persistence diagrams* in topological data analysis we consider X as the space of m unordered points in $\mathbb{R}^2$ under the L_∞-bottleneck distance. Intuitively, the bottleneck distance is defined as the length (in L_∞-distance) of the longest edge in a perfect matching minimizing said length. Computing such a bottleneck distance is another classical problem in computational geometry [3, 7, 10, 11]. The main difference of our setting to persistence diagrams is that the latter contain infinitely many points on the so-called diagonal. Nevertheless, we hope that some of our ideas might be used to construct k-centers for persistence diagrams, at least under some additional assumptions. Such k-centers could be interesting for example for clustering persistence diagrams, a topic that has recently gained attention [4, 5, 12, 13, 15, 17].

In order to illustrate the problem, we give another interpretation in terms of supplying restaurant chains with supermarket chains, which we call the *Restaurant Supply Problem*: In a city, for example Manhattan, there are n different restaurant chains that have m stores each. Their supply is secured by k supermarket chains that also have m stores each. All restaurants that belong to the same chain get their supply from the same supermarket chain. However, each restaurant within one chain gets it from a different store. This means that as soon as a supermarket chain gets chosen by a restaurant chain, each store gets matched to one specific restaurant that it supplies. We are interested in the number of supermarket chains needed to satisfy all restaurant chains. Formally, we define the following:

▶ **Definition 1.** Let $R_1 = (r_1^1, r_1^2, \dots, r_1^m), \dots, R_n = (r_n^1, \dots r_n^m)$ be n restaurant chains with m stores each in some metric space X. We say that a supermarket chain consisting of m supermarkets $s_1, \dots, s_m$ *δ-satisfies* (or equivalently just *satisfies*) a set of restaurant chains $R_a, \dots, R_b$ if there exists a relabeling of the stores in each R_i such that the distance between s_j and r_i^j is at most δ for each i and j. More generally, we say that k supermarket chains δ-satisfy the n restaurant chains $R_1 \dots, R_n$ if $R_1, \dots, R_n$ can be partitioned into k subfamilies, each of which can be δ-satisfied by a single supermarket chain.

Of course, if the restaurant chains are operating sufficiently far from each other, then any supermarket chain can only satisfy one restaurant chain, so in general there are instances where we need n supermarket chains. For this reason, we focus on the case where the restaurant chains are in actual competition and operate close to each other. More specifically, we will assume that any h restaurant chains can be satisfied by a single supermarket chain. This assumption can make a big difference: consider the case where each restaurant chain only has a single store, and assume that distances are measured in Euclidean distance. Then a single supermarket satisfies some h restaurants whenever it lies in the intersection of the disks with radius δ centered at the locations of the restaurants. As these disks are convex, it follows from Helly's theorem that if any 3 of them have a common intersection, then all of them do. In other words, if any 3 restaurants can be satisfied by a single supermarket, then all restaurants can be satisfied by a single supermarket.

The Restaurant Supply Problem is now the optimization problem of minimizing the number of supermarket chains under this assumption. In the rest of this manuscript, we will restrict our attention to cities that, like Manhattan or large parts of Barcelona, are laid out on a square grid and thus distances are measured differently[1]. More formally, we define the distance between two points in the plane as the L_∞-distance. Then the disks of radius δ are axis-aligned squares and from Helly's theorem for boxes we get an even stronger statement for chains with only one store: if any two restaurants can be satisfied by a single supermarket, then all of them can. For this reason, we set $h = 2$ for the rest of this manuscript, that is, we assume that any two restaurant chains can be satisfied by a single supermarket chain. All omitted proofs can be found in the full version [1].

2 An upper bound

▶ **Theorem 2.** *Let $\{r_1^1, \dots, r_1^m\}, \dots, \{r_n^1, \dots, r_n^m\}$ be n restaurant chains with m stores each, such that any two of them can be satisfied by a single supermarket chain. Then all of them can be satisfied by $k = m!$ supermarket chains.*

For each restaurant chain, we consider its m stores and annotate them with the numbers $1, \dots, m$ from left to right. Now consider the permutation π obtained when enumerating the points of that restaurant chain from bottom to top. If any two points have the same x- or y-coordinate, we break ties lexicographically. We claim that all restaurant chains from the same permutation can be satisfied by just one supermarket chain.

▶ **Lemma 3.** *For two restaurants r_1, r_2 annotated with the same index from chains R_1, R_2 in the same permutation, their distance is at most 2δ.*

[1] Formally speaking, in such cities distances are measured in the L_1-metric. However, as we consider only metric balls in our arguments, and as metric balls are squares in the plane under both metrics, in order to be consistent with our motivation from persistence diagrams, we choose to work with the L_∞-metric.

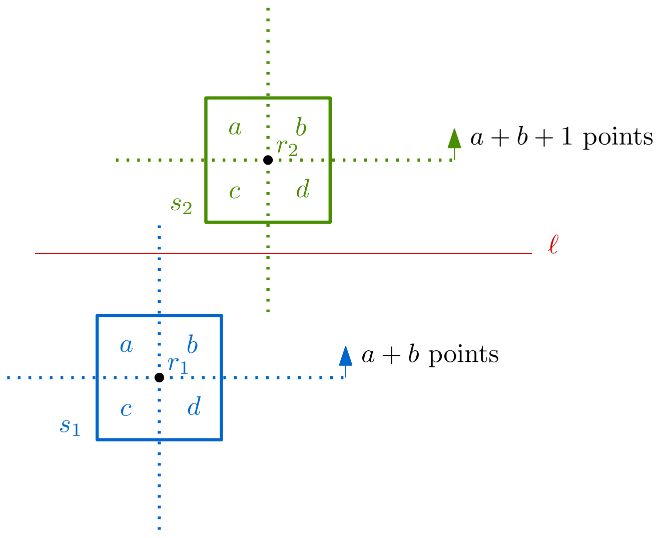


Figure 1 Two restaurants annotated with the same index from chains in the same permutation split the plane into quadrants with the same number of restaurants.

Proof. The restaurants r_1, r_2 split the plane into four quadrants each, see Figure 1. Since both stores are annotated with the same index and belong to chains from the same permutation, their quadrants contain the same number of points. Let a be the number of points in the top-left, b in the top-right, c in the bottom-left and d in the bottom-right quadrant. Furthermore, let s_1 and s_2 be the δ-balls (which are squares in L_∞) around r_1 and r_2. For sake of contradiction, assume that they do not intersect. By the separation theorem, there exists either a horizontal or vertical separation line between them. Without loss of generality consider the case of a horizontal separation line ℓ and let r_2 be above r_1.
By the Helly-like assumption of our theorem, R_1, R_2 can be satisfied using only one supermarket chain. Thus, their stores can be matched in a way such that the distance of the longest matching edge is at most 2δ. There are at least $a+b+1$ restaurants in R_2 (the upper two quadrants plus r_2 itself) that can in such a matching only be matched to restaurants strictly above ℓ. However, R_1 can contain at most $a+b$ such restaurants. By contradiction, s_1 and s_2 intersect. ◀

Proof of Theorem 2. By Lemma 3 any two stores annotated with the same index from chains in the same permutation are at most distance 2δ apart. In other words, their δ-balls intersect, and thus, the statement follows from Helly's theorem for boxes. ◀

3 A lower bound

Unfortunately, the bound proven in the previous section has a superexponential dependence on the number of stores per chain m. As it turns out, no subexponential bound exists.

▶ **Theorem 4.** *There exists a placement of n restaurant chains $\{r_1^1, \dots, r_1^m\}, \dots, \{r_n^1, \dots, r_n^m\}$, with m stores each, such that any two can be satisfied by a single supermarket chain, but satisfying all n chains requires $k \geq \min(n, \lfloor e^{m/(27+\varepsilon)} \rfloor)/2$ supermarket chains, for any constant $\varepsilon > 0$.*

The proof is based on the following construction, an example where a single supermarket chain does not suffice. Consider six points arranged on a circle such that any two squares of

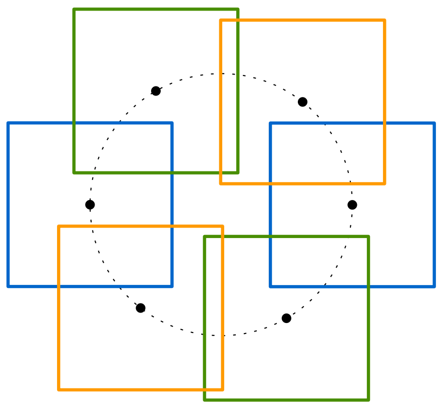


Figure 2 A city with 3 restaurant chains, each consisting of 2 stores placed on antipodal points.

radius δ centered at the points intersect if and only if the points are neighbors on the circle. We refer to this arrangement as a city; an example is illustrated in Figure 2.

Sketch of proof. Consider many cities sufficiently far away from each other. In each city, each restaurant chain constructs independently and uniformly at random two stores on antipodal points. Applying the probabilistic method to this procedure shows that, given the number of cities is sufficiently large, there will, for any restaurant chain triple exist one city where these three chains occupy distinct placements within this city. Consequently, a single supermarket chain cannot satisfy these three restaurant chains. ◀

4 Computational complexity

In this section we analyze the computational complexity of deciding whether all restaurant chains can be satisfied by a single supermarket chain. In both our results it will be helpful to take a graph theoretic viewpoint on the problem: we define the *restaurant graph* G whose vertex set is the set of all restaurant stores, and where two stores r_a^b and r_x^y of different chains are connected whenever their δ-balls intersect. The vertex set of this graph partitions naturally into m-sets of vertices that correspond to the m stores of a restaurant chain. We interpret the different chains as colors and say that a clique C in G is a *colorful clique* if it contains exactly one vertex of each m-set associated to a restaurant chain.

▶ **Lemma 5.** *Let $\{r_1^1, \ldots, r_1^m\}, \ldots, \{r_n^1, \ldots, r_n^m\}$ be n restaurant chains with m stores each. Then all of them can be satisfied by a single supermarket chain if and only if the vertex set of the restaurant graph G can be partitioned into m colorful cliques.*

Proof. Assume there is a placement of the m supermarket stores $s_1, \ldots, s_m$ such that each restaurant r_j^i is satisfied by the store s_i. In particular, any two vertices r_a^i and r_b^i are connected and we can thus partition the vertex set of G into m colorful cliques. On the other hand if such a partition into colorful cliques exists, then for any two stores in this clique their δ-balls intersect. Thus, by Helly's theorem for boxes, all of them intersect, which means that we can place a supermarket store in this intersection to satisfy all the stores in the clique. ◀

The following can be proven using standard methods.

▶ **Theorem 6.** *Let $\{r_1^1, r_1^2\}, \ldots, \{r_n^1, r_n^2\}$ be n restaurant chains, each having two stores, such that any two of them can be satisfied by a single supermarket chain. Then we can decide in time $O(n^3)$ whether all of them can be satisfied with a single supermarket chain.*

Let us mention here that we did not try to optimize the runtime and it is thus likely that it can still be improved. Further, it is an interesting problem whether for a larger constant number of stores we can still solve the problem in polynomial time.

▶ Question 7. Given n restaurant chains, each having m stores, where m is a constant, such that any two of them can be satisfied by a single supermarket chain, is there a polynomial time algorithm to decide whether all of them can be satisfied by a single supermarket chain?

On the other hand, if the number of stores is unbounded, then the problem becomes NP-complete, even if there are only 3 restaurant chains.

▶ **Theorem 8.** *Let $\{r_1^1, \dots, r_1^m\}, \{r_2^1, \dots, r_2^m\}, \{r_3^1, \dots, r_3^m\}$ be 3 restaurant chains with m stores each, such that any two of them can be satisfied by a single supermarket chain. Then it is NP-complete to decide whether all of them can be satisfied with a single supermarket chain.*

Sketch of proof. The containment in NP follows from Lemma 5. For NP-hardness, we use a reduction from planar 3-SAT [14]. We describe our construction in terms of the δ-balls and give a different color to each chain. Thus, given a planar SAT formula φ we place some number n of blue squares, n green squares and n orange squares such that we can partition them into colorful triples with common intersections if and only if φ is satisfiable.

For each variable in the formula φ we arrange green and blue squares in a cycle, see Figure 3. There are exactly two types of placements of supermarkets to satisfy the involved restaurants, which we interpret as “true” and “false”, respectively.

For the clauses we use squares of the third color, orange. We place an orange square o_1 in such a way that it intersects exactly the regions a_1, a_2 and a_3 corresponding to setting a literal to “true”, see Figure 4. We place two more orange squares o_2, intersecting a_1 and a_2 as well as the intersections on the variable gadgets before them, and o_3, intersecting a_2 and a_3 as well as the intersections after them. Note that, locally, this configuration can be satisfied if any only if at least one literal is set to “true”.

We link the variable and clause gadgets according to a plane embedding of the variable-clause graph, see Figure 5. It remains to place more orange squares such that any two color classes can be satisfied: we add one more orange square for each variable-clause incidence (see Figure 6) and add orange copies of almost all blue squares, except the first two blue squares of any variable-clause incidence (denoted b_i and b_{i-1} in Figure 6). It now follows from the construction that any two color classes can be satisfied with a single supermarket chain, and that all three can be satisfied if and only if φ is satisfiable. ◀

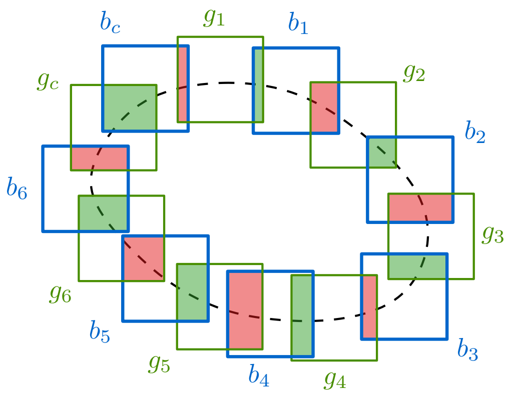


Figure 3 Placing supermarkets in the green (red) regions sets the variable to “true” (“false”).

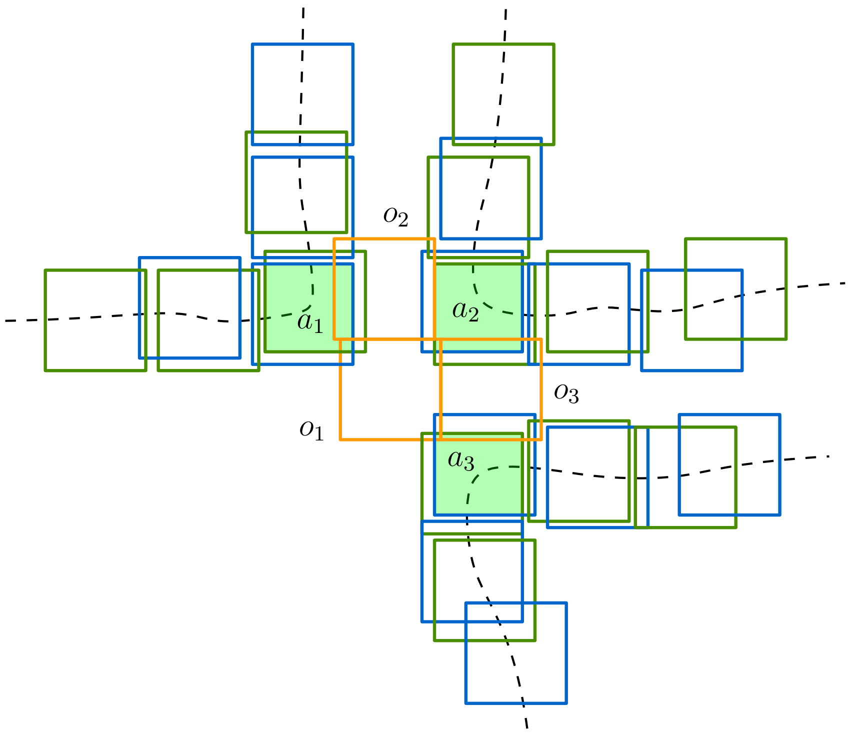


Figure 4 A clause gadget.

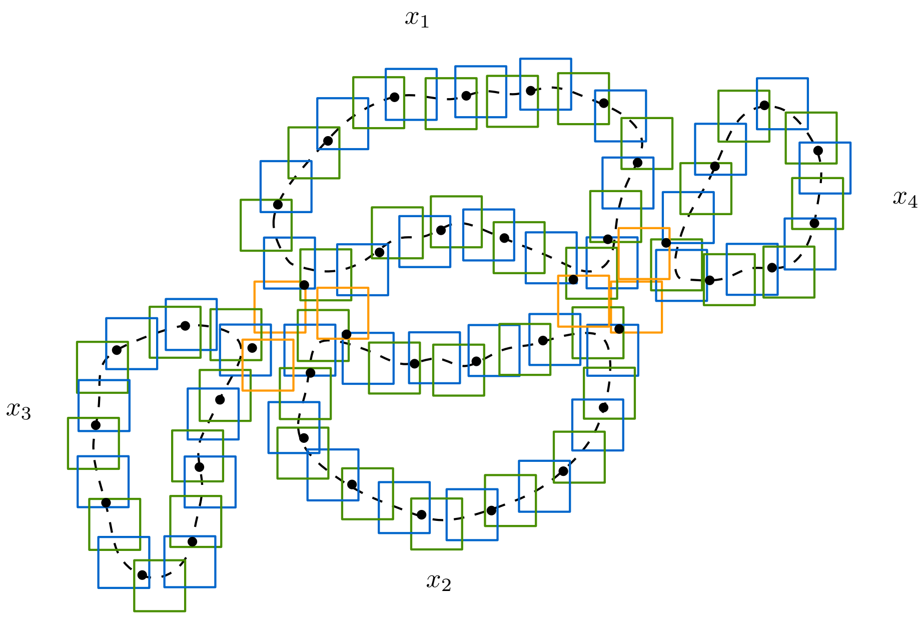


Figure 5 All gadgets for the (very small) formula $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$. The black points describe a placement of supermarkets corresponding to setting all variables to "true".

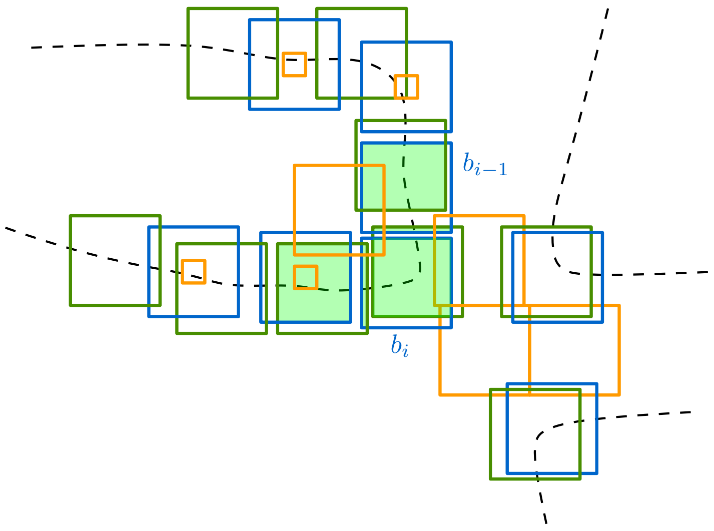


Figure 6 The placement of the remaining orange squares. The orange squares that are copies of blue squares are drawn small.

Acknowledgments. This work was initiated during the 21st Gremo's Workshop on Open Problems in Pura, Switzerland. The authors would like to thank the other participants for the lively discussions. The first two authors were supported by the German Academic Scholarship Foundation. The second author was further supported by a fellowship of the German Academic Exchange Service (DAAD) and the Cusanuswerk.

References

1 Mats Bierwirth, Julia Hütte, Patrick Schnider, and Bettina Speckmann. Bounds for k-centers of point sets under l_∞-bottleneck distance, 2025. URL: `https://arxiv.org/abs/2503.02715`, `arXiv:2503.02715`.

2 Mihai Bundefineddoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, page 250–257, New York, NY, USA, 2002. Association for Computing Machinery. `doi:10.1145/509907.509947`.

3 Rainer E. Burkard and Ulrich Derigs. *The Bottleneck Matching Problem*, pages 60–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 1980. `doi:10.1007/978-3-642-51576-7_5`.

4 Yueqi Cao, Prudence Leung, and Anthea Monod. k-means clustering for persistent homology. *Advances in Data Analysis and Classification*, pages 1–25, 2024.

5 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pages 263–271, 2005.

6 Graham Cormode and Andrew McGregor. Approximation algorithms for clustering uncertain data. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '08, page 191–200, New York, NY, USA, 2008. Association for Computing Machinery. `doi:10.1145/1376916.1376944`.

7 Alon Efrat and Alon Itai. Improvements on bottleneck matching and related problems using geometry. In *Proceedings of the twelfth annual symposium on Computational geometry*, pages 301–310, 1996.

8 Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second*

International Conference on Knowledge Discovery and Data Mining, KDD'96, page 226–231. AAAI Press, 1996.

9 Gabriel Y. Handler and Pitu B. Mirchandani. Location on networks : theory and algorithms. 1979. URL: https://api.semanticscholar.org/CorpusID:54147417.

10 A. Karim Abu-Affash, Paz Carmi, Matthew J. Katz, and Yohai Trabelsi. Bottleneck non-crossing matching in the plane. *Computational Geometry*, 47(3, Part A):447–457, 2014. URL: https://www.sciencedirect.com/science/article/pii/S0925772113001260, doi:10.1016/j.comgeo.2013.10.005.

11 Matthew J. Katz and Micha Sharir. Bottleneck matching in the plane. *Computational Geometry*, 112:101986, 2023. URL: https://www.sciencedirect.com/science/article/pii/S0925772123000068, doi:10.1016/j.comgeo.2023.101986.

12 Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams, 2017.

13 Théo Lacombe, Marco Cuturi, and Steve Oudot. Large scale computation of means and clusters for persistence diagrams using optimal transport. *Advances in Neural Information Processing Systems*, 31, 2018.

14 David Lichtenstein. Planar formulae and their uses. *SIAM journal on computing*, 11(2):329–343, 1982.

15 Andrew Marchese, Vasileios Maroulas, and Josh Mike. K- means clustering on the space of persistence diagrams. In *Wavelets and Sparsity XVII*, volume 10394, pages 218–227. SPIE, 2017.

16 Nimrod Megiddo and Kenneth J. Supowit. On the complexity of some common geometric location problems. *SIAM Journal on Computing*, 13(1):182–196, 1984. arXiv:https://doi.org/10.1137/0213014, doi:10.1137/0213014.

17 Jules Vidal, Joseph Budin, and Julien Tierny. Progressive Wasserstein barycenters of persistence diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):151–161, 2020. doi:10.1109/TVCG.2019.2934256.

Mutations and Euclideaness in pseudosphere arrangements

Michael Wilhelmi[1]

1 **FernUniversitaet in Hagen**
`mail@michaelwilhelmi.de`

Abstract

An oriented matroid program $(\mathcal{O}, g, f)$ is an oriented matroid (OM) (the set of sign-vectors of the cells of an oriented pseudosphere arrangement) together with two elements (pseudospheres) g and f. We call an oriented matroid program *Euclidean* if to each cocircuit (which is a vertex of the arrangement) there is an extension parallel to f (with respect to g) going through the cocircuit. We call an oriented matroid *Mandel* if it has an extension in general position which makes all programs with that extension Euclidean. A simplicial region in the arrangement is called a *mutation*.

Let L be the minimum number of mutations adjacent to an element in an OM or in a class of OMs. We call an OM *Las Vergnas* if $L > 0$. If $\mathfrak{O}_{\text{property}}$ is the class of OMs having a certain property, it holds $\mathfrak{O} \supset \mathfrak{O}_{\text{Las Vergnas}} \supset \mathfrak{O}_{\text{Mandel}} \supset \mathfrak{O}_{\text{Euclidean}} \supset \mathfrak{O}_{\text{realizable}}$. All these inclusions are proper and we give explicit proofs and examples for the parts of this chain that were not known.

For realizable hyperplane arrangements of rank r we have $L = r$ which was proved by Shannon. Under the assumption that a (modified) intersection property holds we give an analogue to Shannon's proof and show that uniform rank-4 Euclidean oriented matroids with that property have $L = 4$. Using the fact that the lexicographic extension creates and destroys certain mutations, we show that for Euclidean oriented matroids, $L \geq 3$ holds.

We can reverse a mutation of an oriented matroid and obtain a new OM. This is called a *mutation-flip*. We prove that Euclideaness is preserved after a certain type of mutation-flips which yields that a path in the mutation-graph from a Euclidean oriented matroid to a totally non-Euclidean oriented matroid (which has no Euclidean oriented matroid programs) must have at least three mutation-flips. Finally, a minimal non-Euclidean or rank-4 uniform oriented matroid is Mandel if it is connected to a Euclidean oriented matroid via one mutation-flip. Hence, we get many examples for Non-Euclidean but Mandel oriented matroids and have $L \leq 3$ for those of rank 4.

Related Version https://arxiv.org/abs/2501.12951

1 Introduction

A central arrangement of hyperplanes in $\mathbb{R}^r$ is a finite family $\mathcal{A} = (H_e)_{e \in E}$ of hyperplanes in $\mathbb{R}^r$. The *rank* of the arrangement is the codimension of the intersection of all its hyperplanes. We assume $\bigcap \mathcal{A} = \{0\}$, hence $\mathcal{A}$ has rank r. A hyperplane H_e divides $\mathbb{R}^r$ into two disjoint open halfspaces and itself. We assign an orientation to H_e by designating one halfspace as *positive* (H_e^+) and the other as *negative* (H_e^-). Each point $p \in \mathbb{R}^r$ corresponds to a sign-vector $S \in \{0, +, -\}^E$ such that $S_e = 0, S_e = +$ or $S_e = -$ if p lies in H_e, H_e^+ or in H_e^-. The set of all points sharing the same sign-vector is called a *cell* and we obtain a partition of $\mathbb{R}^r$ into disjoint cells. The *dimension* of a cell is the dimension of its linear hull. A cell of dimension 1 is a *vertex*, of dimension 2 an *edge*, and a full-dimensional cell is a *region*. A region corresponds to a sign-vector that has no zero-entries. If the closure of a region is a simplex, we call it a *simplicial region* or a *mutation*. We say a hyperplane is *adjacent* to a mutation if it contains a facet of the mutation. There is a famous result by Shannon, see [14], that each hyperplane of an arrangement $\mathcal{A}$ of rank r has r adjacent mutations. We cite here the proof in [2], Lemma 2.1.5, see Figure 1, which uses induction by rank. The first part (see

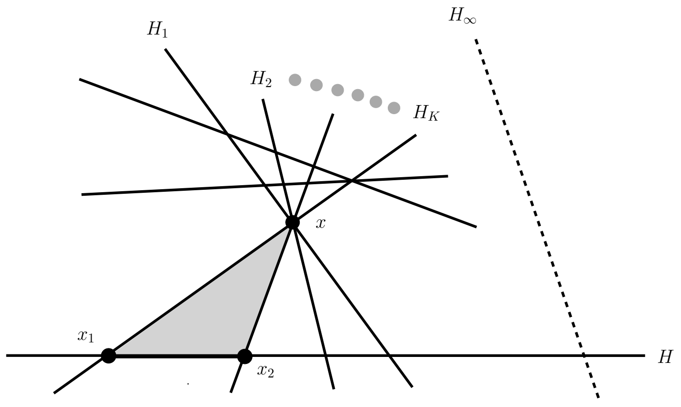


Figure 1 Proof of Shannon's result, from [2], Lemma 2.1.5

also [14]) shows that for any two hyperplanes H and H_∞ in $\mathcal{A}$, there exists a vertex $x \notin H$ in a simplicial region adjacent to H with no vertices in H_∞. Let x be the closest vertex to H not lying in H_∞. Since $\mathcal{A} \cap H = (H_e \cap H)_{e \in E}$ is a hyperplane arrangement of rank $r-1$, it follows by induction, that it has a simplicial region in H outside $H \cap H_\infty$ with vertices $x_1, \ldots, x_{r-1}$. Because x is closest to H, all $[x_i, x]$ are edges, forming a simplicial region containing x outside H_∞. The second part (see also [13]) says, that if we have less than r mutations adjacent to H, we can add a hyperplane H_∞ intersecting all these mutations. There must be at least one more mutation adjacent to H outside H_∞.

If we intersect a central arrangement of hyperplanes in $\mathbb{R}^r$ with the sphere S^{r-1}, we obtain an arrangement of $(r-2)$-subspheres in S^{r-1} while preserving the original sign-vector information. If we now consider subsets S of S^{r-1} homeomorphic to S^{r-2} (see [6]) and orient them like before, we obtain *arrangements of pseudospheres*. These arrangements again divide the sphere into disjoint cells corresponding to sign-vectors which are the set of *covectors* of an *oriented matroid* (or an *OM*). Sign-vectors corresponding to vertices, edges and regions of the arrangement are called *cocircuits*, *edges* and *topes*. The pseudospheres S_e of the arrangement correspond to *elements* of the *groundset* E of the OM. We always assume $\bigcap_{e \in E} S_e = \emptyset$. Then, the OM has *rank* r and *corank* $|E| - r$. If the deletion of an element of an OM decreases the rank, we call it a *coloop*. In the following, we only consider OMs without coloops. Oriented matroids can also be defined as a set of covectors in a purely axiomatic way. Many of these OMs cannot be 'stretched', they correspond to pseudosphere- but not to sphere-arrangements. We call them *non-realizable*. Shannon's result holds for realizable OMs (so also for OMs with rank or corank ≤ 2) and for rank-3 OMs (which are also called *pseudoline arrangements*). Each pseudoline has at least 3 adjacent mutations (this was shown by Levi in [10]). You can see this in Figure 2. The mutations are triangles here.

In rank 4, things get more complicated. In the 80's, Roudneff, Sturmfels and Altshuler found a rank-4 OM with 8 elements called the $RS(8)$, see [13], that has an element being only adjacent to 3 mutations, see Figure 3, the red triangles on the button element. The conjecture (made by Las Vergnas in [9]) that each oriented matroid has at least one mutation is still open today. We call OMs *Las Vergnas*, if each element has at least one adjacent mutation. In the 90's and early 2000's, Richter-Gebert, Bokowski, Rohlfs, and Hall, see [12], [4], [5], found OMs built on the $RS(8)$ with mutation-free elements, hence being not Las Vergnas. We later use the OM that Richter-Gebert found, which is called the $R(20)$.

The vertices and edges of a central hyperplane arrangement form a graph. Analogously, the *cocircuit graph* of an oriented matroid has the cocircuits as its vertices and the OM's

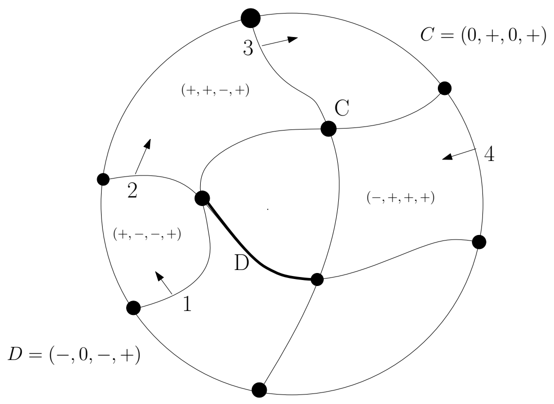


Figure 2 An oriented matroid of rank 3 with its covectors, C is a cocircuit, D is an edge, we look at the 2-sphere from above.

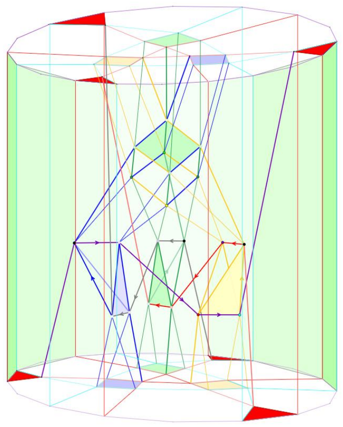


Figure 3 from [4], the $RS(8)$, a rank-4 OM. We look at a 3-sphere. See [4], page 289 for details.

edges as its edges. An OM $\mathcal{O}$ together with two elements g (the *infinity*) and f (the *target function*) yields an *oriented matroid program* (*OMP*), written as $(\mathcal{O}, g, f)$. We call an OMP *Euclidean* if, for each cocircuit X with $X_g = +$, there exist an extension of the OMP by a single element f' through X, parallel to f with respect to g (meaning $X_{f'} = 0$ and $Y_f = Y_{f'}$ for all cocircuits Y with $Y_g = 0$). An OM is *Euclidean* if all its programs are. In $(\mathcal{O}, g, f)$ we can direct the edges in the cocircuit graph between cocircuits with $g = +$. We go along an edge (X, Y) from X to Y and continue until we reach the element g in a cocircuit Z. If Z lies in the positive (negative) halfspace of f, the direction *increases* (*decreases*), and the edge is directed from X to Y (from Y to X), see Figure 4. A famous result by Edmonds, Fukuda and Mandel, see [11], states that OMPs are Euclidean iff their cocircuit graph has no directed cycles. OMs of rank ≤ 3 or corank ≤ 3 are always Euclidean. In rank 4, this is not always the case. The $RS(8)$ is one such counterexample, as shown by the cycle in Figure 3.

2 Mandel OMs are Las Vergnas, but not vice versa

In his thesis [11], Arnaldo Mandel made a“wishful thinking” by conjecturing that every OM has an extension g in general position, so that all its programs using g as infinity are Euclidean. We call OMs with that property *Mandel*. The next theorem has already been

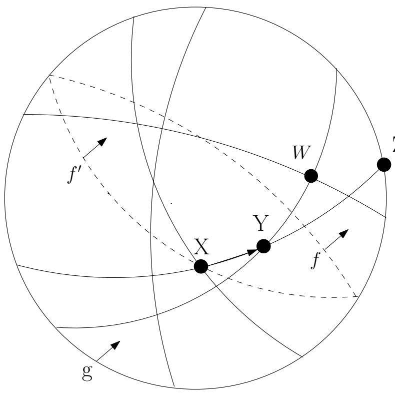


Figure 4 A Euclidean oriented matroid program. f' goes through X parallel to f wrt. g. From the edge (X, Y), we reach g in the cocircuit Z with $Z_f = +$. It has increasing direction (from X to Y). All edges from W going away from f are increasing. W lies in a triangle adjacent to f.

shown in [11] and in [8]. We show it again by highlighting the similarity to Shannon's proof.

▶ **Theorem 2.1.** *Mandel oriented matroids are Las Vergnas.*

If we want to show that an element f has an adjacent mutation, we use f as target function in the OMP $(\mathcal{O}, g, f)$, where g is the 'Mandel' extension. We go on like in the first part of Shannon's proof. Even if we have no notion of distance in OMs anymore, the cocircuit graph of $(\mathcal{O}, g, f)$ is directed because the program is Euclidean and there is a cocircuit W in that graph whose predecessors are only cocircuits lying on f. Then W has the same function like the vertex x in Shannon's proof, see again Figure 4. We know today that non-Las Vergnas OMs exist, hence Mandel's conjecture turned out to be wrong. We come to a new result.

▶ **Theorem 2.2** ([15], Theorem 10 and [7], Theorem 1.2)**.** *There are Las Vergnas oriented matroids that are not Mandel. Euclidean oriented matroids are Mandel.*

To prove it we need the lexicographic extension of an OM, which adds an element f' in general position 'very close' to an element f. The two elements f and f' are *inseparable* in the OM, which means that no cocircuits lie between them. This extension destroys some

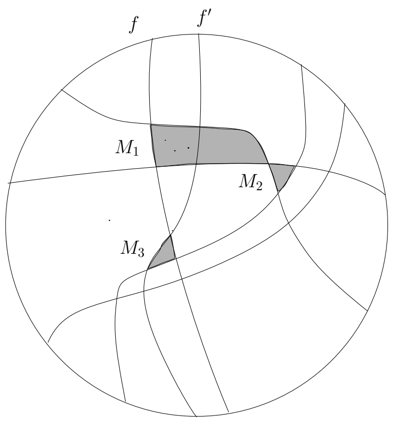


Figure 5 The lexicographic extension added the element f'. The elements f and f' are inseparable. Mutation M_1 is cutted in half by f', mutation M_2 remains unchanged and mutation M_3 is new.

mutations adjacent to f, leaves non-adjacent mutations as they are and creates at least one new mutation, see Figure 5. With this insight we can construct an OM being Las Vergnas, but not Mandel. The $R(20)$, which is neither Las Vergnas nor Mandel, has only one mutation-free element f. We extend it lexicographically with an element f' inseparable to f. Now, f and f' are adjacent to a new mutation and all other mutations stay like before. Hence, the extended OM is Las Vergnas but not Mandel because an OM remains Mandel if we delete an element of it (except in some trivial cases, see [15], Theorem 6). For the second statement of the theorem, we use our main result of [7] (joint work with W. Hochstättler): The lexicographic extension of a Euclidean oriented matroid remains Euclidean. Hence, any such extension is the desired extension in general position for a Euclidean OM being Mandel.

3 New results on the number of mutations in OMs

If L is the minimal number of mutations adjacent to an element in a class of OMs, we show:

▶ **Theorem 3.1** ([15], Theorem 13). *For Euclidean oriented matroids (without coloops) of rank $r > 3$ and corank ≥ 3, it holds that $L \geq 3$.*

To show this, for each element f, we choose an element g such that f and g are separable. (The case that there is no such g can be excluded using non-trivial results about *binary* OMs from matroid theory, we refer to [15], Lemma 1 and before for the details.) Because g and f are separable, we have $g = +$ cocircuits with $f = +$ and $f = -$. Hence, because $(\mathcal{O}, g, f)$ is Euclidean we get one mutation on each side of f. No matter where these two mutations are located, we always find a lexicographic extension f' that destroys both. The program $(\mathcal{O}, f', f)$ remains Euclidean and we obtain a third mutation.

In general, a lexicographic extension cannot destroy more than two mutations. For more, we need a stronger property than Euclideaness called the *intersection property* IP_1, see [1].

▶ **Definition 3.2.** An oriented matroid of rank r has the intersection property IP_1 if, for each set of $r-1$ cocircuits, there exists a (non-trivial) extension going through these cocircuits.

Such an extension (and assuming it preserves Euclideaness) yields an analogue to the second part of the proof of Shannon's result. If there are $r-1$ mutations, we use the extension intersecting $r-1$ cocircuits, one from each mutation, then slightly perturb it to destroy all mutations (preserving Euclideaness), gaining one additional mutation as before. We could only prove a special case (an OM is *uniform* iff all elements lie in general position):

▶ **Theorem 3.3** ([15], Theorem 14). *For uniform oriented matroids of rank 4 and corank ≥ 4 that satisfy the IP_1, where the extensions provided by the IP_1 can be chosen to be Euclidean, $L = 4$ holds.*

Since the extension yielded by the IP_1 can intersect more than the $r-1$ cocircuits, extensive case-checking was required in the proof, making generalization to higher ranks difficult.

4 OMs with a Euclidean mutant are Mandel

In uniform oriented matroids, each mutation corresponds to a *mutation-flip*. We move one pseudosphere f of the mutation across the cocircuit X of the mutation that is not adjacent to f, as shown in Figure 6, obtaining a new OM called a *mutant*. The *mutation-graph* has uniform oriented matroids as vertices, with mutants connected by edges. Such a mutation-flip does not affect the Euclideaness of a program $(\mathcal{O}, g, f)$ if f is adjacent to the mutation and g is not, see Figure 6. It is obvious that cocircuits with $g = 0$ remain unchanged after the flip,

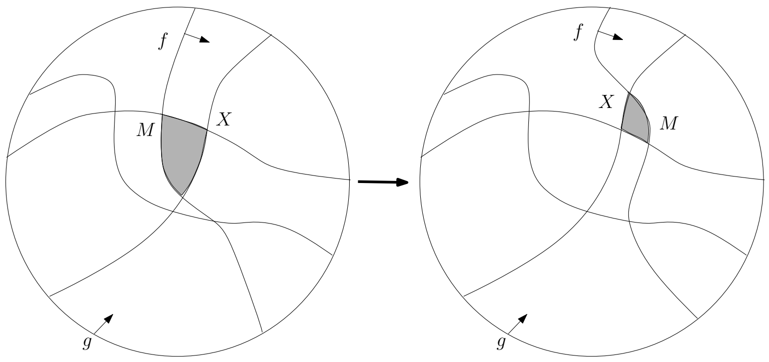


Figure 6 Two oriented matroids connected by a mutation-flip. Element f is adjacent to the mutation, element g not. All edges from f to the cocircuit X have the same direction in both OMs.

hence, all edges outside of the mutation retain their directions. Additionally, the cocircuit X cannot be part of a cycle, all edges from f to X have the same direction, and such a cycle can never contain cocircuits with $f = 0$.

We present two applications of these results. First, we again consider Richter-Gebert's $R(20)$. It is a uniform *totally non-Euclidean* OM, meaning it has no Euclidean oriented matroid programs. (We verified this with computer help but will prove it by hand in an upcoming paper.) Each path from a Euclidean OM to the $R(20)$ in the mutation-graph must include at least three mutation-flips, see Corollary 4 in [15]. After the first flip of mutation M_1 (where $M_1 = \{e_1, \dots, e_r\}$ is the set of elements adjacent to the mutation), programs $(\mathcal{O}, g, f)$ with $(g, f) \in M_1 \times E \setminus M_1$ remain Euclidean. After the second flip of mutation M_2, programs $(\mathcal{O}, g, f)$ with $((g, f) \in M_1 \setminus M_2 \times M_2 \setminus M_1)$ still remain Euclidean. Thus, only after three flips is it possible for all Euclidean oriented matroid programs to become non-Euclidean. The second application is the following theorem.

▶ **Theorem 4.1** ([15], Theorem 17, Theorem 18). *A uniform oriented matroid $\mathcal{O}$ of rank 4 (or a minimal non-Euclidean uniform oriented matroid) is Mandel if it has a Euclidean mutant.*

The proof proceeds as follows, see Figure 7. First, the mutant $\mathcal{O}_M$ is Euclidean. Flipping $\mathcal{O}_M$ back to $\mathcal{O}$ preserves the Euclideaness of the programs $(\mathcal{O}, g, f)$ if $f \in M$ and $g \notin M$. We extend $\mathcal{O}$ lexicographically and obtain $\mathcal{O}_{f'}$, which has a new mutation $M' = M \setminus f \cup f'$. The programs $(\mathcal{O}_{f'}, g, f')$ and $(\mathcal{O}_{f'}, f, f')$ remain Euclidean. We flip $\mathcal{O}_{f'}$ on the mutation M' and obtain $\mathcal{O}_{f',M'}$. Again, the programs $(\mathcal{O}_{f',M'}, g, f')$ and $(\mathcal{O}_{f',M'}, f, f')$ remain Euclidean. If we have $e \in M \setminus f$, we start with $\mathcal{O}_M$. The program $(\mathcal{O}_M, e, f)$ is Euclidean. We extend $\mathcal{O}_M$ lexicographically with an extension 'reversed' to the first one and obtain $\mathcal{O}_{M,f'}$. The program $(\mathcal{O}_{M,f'}, e, f)$ remains Euclidean. We again have the mutation M' in $\mathcal{O}_{M,f'}$. We flip $\mathcal{O}_{M,f'}$ again on the mutation M' and obtain $\mathcal{O}_{M,f',M'}$. Because $f \notin M'$ and $e \in M'$, the program $(\mathcal{O}_{M,f',M'}, e, f)$ remains Euclidean. The mapping that maps f to f' and vice versa, and is the identity for the rest, yields an isomorphism between $\mathcal{O}_{M,f',M'}$ and $\mathcal{O}_{f',M'}$. Hence, the programs $(\mathcal{O}_{f',M'}, e, f')$ are Euclidean for all $e \in E$, and we have $\mathcal{O} = \mathcal{O}_{f',M'} \setminus f'$. That means that $\mathcal{O}$ is Mandel. In Lemma 25 of [15], where the main proof of Theorem 4.1 is given, we assumed that $\mathcal{O}/f$ (see [2], Lemma 4.1.8) must be Euclidean. Removing this assumption seems difficult but not impossible and would generalize Theorem 4.1 to higher ranks.

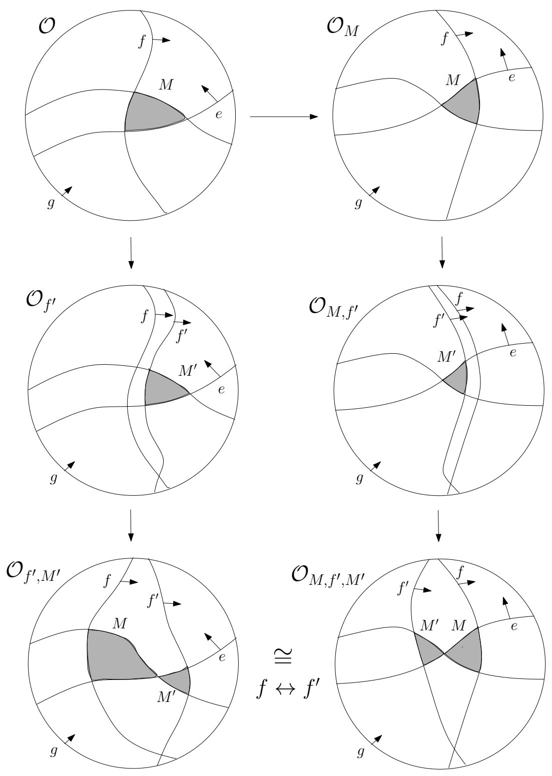


Figure 7 Constellation for the proof of Theorem 4.1

5 Conclusion

All 8-element rank-4 OMs, including $RS(8)$, have a Euclidean mutant, see [3], making them Mandel, even if they are non-Euclidean. Because $RS(8)$ has an element with only 3 adjacent mutations, we obtain $L \leq 3$ for the class of Mandel OMs of rank 4. We get a full chain of strict inclusions. If $\mathfrak{O}_{\text{property}}$ is the class of OMs having a certain property, it holds

▶ **Theorem 5.1.** $\mathfrak{O} \supset \mathfrak{O}_{Las\ Vergnas} \supset \mathfrak{O}_{Mandel} \supset \mathfrak{O}_{Euclidean} \supset \mathfrak{O}_{realizable}$.

Regarding the strictness of the last inclusion, note that while some rank-3 OMs (pseudoline arrangements) are non-realizable, they are always Euclidean. Finally, considering only OMs without coloops and specifying that they have rank r and corank cr, we have:

▶ **Theorem 5.2.** *1.* $L(\mathfrak{O}) = 0$ *(Richter-Gebert, Bokowski/Rohlfs, Hall).*
2. $L(\mathfrak{O}) = r$ *if* $r \leq 3$ *(Levi) and* $L(\mathfrak{O}_{realizable}) = r$ *(Shannon).*
3. $1 \leq L(\mathfrak{O}_{Mandel})$ *(Mandel, Knauer/Marc) and* $L(\mathfrak{O}_{Mandel}) \leq 3$ *if* $r, cr \geq 4$ *(new result).*
4. $3 \leq L(\mathfrak{O}_{Euclidean}) \leq r$ *if* $r > 3$*,* $cr \geq 3$ *(new result).*
5. $L(\mathfrak{O}_{IP_1,uniform}) = 4$ *if* $r = 4$ *and* $cr \geq 4$ *(new results).*

References

1 Achim Bachem and Alfred Wanka. Euclidean intersection properties. *Journal of Combinatorial Theory, Series B*, 47(1):10–19, 1989. URL: `https://www.sciencedirect.com/science/article/pii/0095895689900610`, `doi:https://doi.org/10.1016/0095-8956(89)90061-0`.

2 Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Gunter M. Ziegler. *Oriented Matroids*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2 edition, 1999.

3 Jürgen Bokowski and Jürgen Richter-Gebert. *On the classification of non-realizable oriented matroids*. Technische Hochsch., Fachbereich Mathematik, 1990.

4 Jürgen Bokowski and Holger Rohlfs. On a mutation problem for oriented matroids. *European Journal of Combinatorics*, 22(5):617–626, 2001. URL: `https://www.sciencedirect.com/science/article/pii/S0195669800904839`, `doi:https://doi.org/10.1006/eujc.2000.0483`.

5 Huntington T. Hall. *Counterexamples in Discrete Geometry*. Thesis (Ph.D.)–University of California, Berkeley, 2004. URL: `https://books.google.de/books?id=MypPAQAAMAAJ`.

6 Winfried Hochstättler. Oriented matroids from wild spheres. *Journal of Computational Technologies*, 7:14–24, 2002.

7 Winfried Hochstättler and Michael Wilhelmi. Vertex-shellings of euclidean oriented matroids. `https://arxiv.org/abs/2306.07184v1`, 2023.

8 Kolja Knauer and Tilen Marc. Corners and simpliciality in oriented matroids and partial cubes. *European Journal of Combinatorics*, 112:103714, 2023. URL: `https://www.sciencedirect.com/science/article/pii/S0195669823000318`, `doi:https://doi.org/10.1016/j.ejc.2023.103714`.

9 Michel Las Vergnas. Convexity in oriented matroids. *Journal of Combinatorial Theory, Series B*, 29(2):231–243, 1980. URL: `https://www.sciencedirect.com/science/article/pii/0095895680900829`, `doi:https://doi.org/10.1016/0095-8956(80)90082-9`.

10 Friedrich Levi. Die Teilung der projectiven Ebene durch Gerade oder Pseudogerade. *Sächs. Akad. Wiss. Math.-Phys. Klasse 78*, pages 256–267, 1926.

11 Arnaldo Mandel. *Topology of Oriented Matroids*. Thesis (Ph.D.)–University of Waterloo, 1982.

12 Jürgen Richter-Gebert. Oriented matroids with few mutations. *Discrete & Computational Geometry*, 10(3):251–269, 1993. `doi:10.1007/BF02573980`.

13 Jean-Pierre Roudneff and Bernd Sturmfels. Simplicial cells in arrangements and mutations of oriented matroids. *Geometriae Dedicata*, 27(2):153–170, 1988. `doi:10.1007/BF00151346`.

14 Robert W. Shannon. Simplicial cells in arrangements of hyperplanes. *Geometriae Dedicata*, 8:179–187, 1979. URL: `https://api.semanticscholar.org/CorpusID:119681116`.

15 Michael Wilhelmi. Mutations and (non-)euclideaness in oriented matroids. `https://arxiv.org/abs/2501.12951`, 2025.

Chasing puppies on orthogonal straight-line plane graphs[*]

Johanna Ockenfels[1], Yoshio Okamoto[2], and Patrick Schnider[3]

1 **Department of Computer Science, ETH Zürich, Switzerland**
jockenfels@student.ethz.ch
2 **Department of Computer and Network Engineering, The University of Electro-Communications, Japan**
okamotoy@uec.ac.jp
3 **Department of Mathematics and Computer Science, University of Basel, Switzerland and Department of Computer Science, ETH Zürich, Switzerland**
patrick.schnider@inf.ethz.ch

Abstract

Assume that you have lost your puppy on an embedded graph. You can walk around on the graph and the puppy will run towards you at infinite speed, always locally minimizing the distance to your current position. Is it always possible for you to reunite with the puppy? We show that if the embedded graph is an orthogonal straight-line embedding the answer is yes.

1 Introduction

The puppy chasing problem was proposed by Michael Biro at the 2013 edition of the Canadian Conference on Computational Geometry. While inspired by problems in beacon-base routing and originally illustrated by train tracks and locomotives, like many other problems in discrete and computational geometry it has a dog-related illustration, which was proposed by the authors who solved Biro's original problem [2].

Assume you are walking with your puppy on an embedded graph. Suddenly, you realize that the puppy is not with you anymore. Luckily, you can see each other and the puppy also wants to reunite with you. While the puppy can run infinitely fast, its behavior is very naive: it runs as close to you as possible, always locally improving the distance to you. Is there always a route you can walk, so that the puppy will eventually reunite with you?

To phrase it in mathematical terms, let G be a finite graph and let $\gamma\colon G \to \mathbb{R}^2$ be a (crossing-free) embedding of the graph. We interpret G as a topological space. Any pair $(x, y) \in G \times G$ then defines locations $h = \gamma(x)$ and $p = \gamma(y)$ in the plane $\mathbb{R}^2$, which we interpret as the positions of the human and the puppy, respectively. Denoting by $d(x, y)$ the Euclidean distance between $h = \gamma(x)$ and $p = \gamma(y)$, we say that a configuration $(x, y) \in G \times G$ is *stable* if there is some $\varepsilon > 0$ such that for all $y' \in G$ with $d(y, y') \leq \varepsilon$ we have $d(x, y') \geq d(x, y)$. In other words, a configuration is stable if the puppy cannot locally decrease its distance to the human. For a non-stable configuration there is thus at least one direction in which the puppy can decrease its distance to the human, in which case it will do so. If there are several such directions, the puppy chooses an arbitrary one. See Figure 1.

If G is a cycle, and the embedding is either smooth or piece-wise linear, it was shown by Abrahamsen, Erickson, Kostitsyna, Löffler, Miltzow, Urhausen, Vermeulen and Viglietta

* This work was initiated during the 21st Gremo's Workshop on Open Problems in Pura, Switzerland. The authors would like to thank the other participants for the lively discussions. The second author acknowledges the support by JSPS KAKENHI Grant Number JP23K10982.

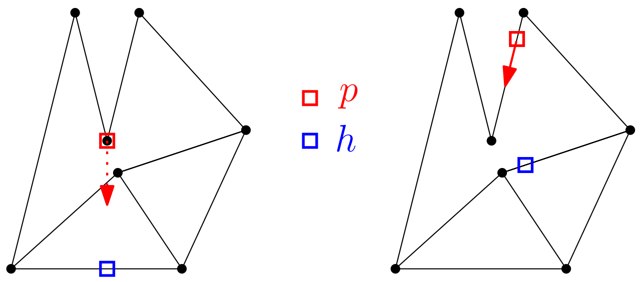


Figure 1 A stable configuration (left), and an unstable one (right).

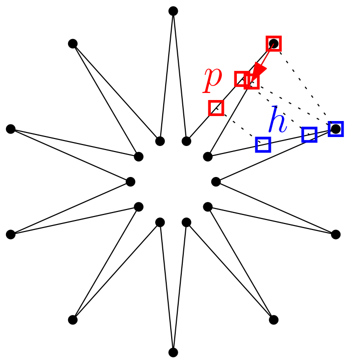


Figure 2 If the human h continues walking clockwise, human and puppy p will never be reunited.

(AEKLMUVV) that the human can always catch the puppy [2], solving Biro's original question. While there are embeddings where the human does not even need to move to catch the puppy, e.g., the unit circle, there are also examples where if moving the wrong way the human will never catch the puppy, see, e.g., Figure 2. If self-intersections are allowed (think of one edge crossing over the other via a bridge), then there are drawings where the human can never catch the puppy. An example of this is the double loop depicted in Figure 3.

It is an interesting open problem to characterize the drawings of a cycle on which the human can always catch the puppy. Towards this, Brunck, Löffler and Silveira have recently shown that the rotation number does not give such a characterization by constructing a curve with rotation number 1 (i.e., the same rotation number as a crossing-free embedding), where the human sometimes cannot catch the puppy [10].

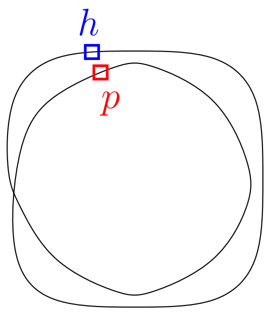
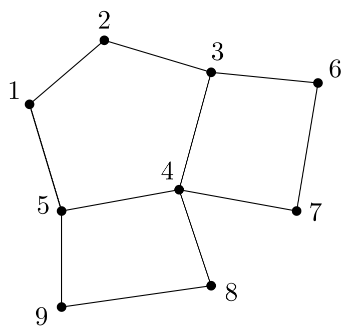


Figure 3 No matter how the human h moves, human and puppy p will never be reunited.



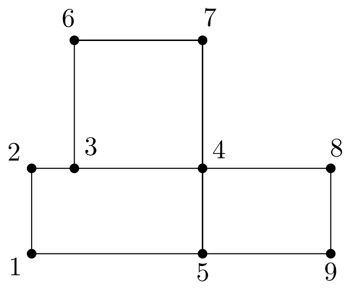


Figure 4 A straight-line embedding (left) and an orthogonal straight-line embedding (right) of the same graph.

1.1 Our results

In this work, we extend Biro's question from embeddings of circles to embeddings of general graphs. Concretely, motivated by the many results on beacon routing and related problems in orthogonal domains, we show the following theorem.

▶ **Theorem 1.1.** *Let G be a finite connected graph and let γ be an orthogonal straight-line embedding of G in the plane. Then, there is a strategy for the human to catch the puppy on γ.*

Here, an orthogonal straight-line embedding is an embedding of the graph where every edge is mapped to either a horizontal or a vertical line segment, see Figure 4 for an example. Note that not every graph admits an orthogonal straight-line embedding, so we are implicitly restricting our attention to those that do. Also note that *orthogonal drawings*, where edges can be drawn as polygonal lines with vertical and horizontal parts can be seen as orthogonal straight-line embeddings of a subdivision of G, so Theorem 1.1 immediately extends to orthogonal drawings.

Unlike the proof of AEKLMUVV that uses topological methods to show the existence of a strategy, we give a recursive strategy that is simple to describe. For orthogonal polygons (that is, orthogonal straight-line embeddings of a circle), AEKLMUVV also give a simple strategy (see [2], Theorem 2), which differs from our strategy. On the other hand, while we conjecture that Theorem 1.1 should hold for any straight-line embedding of a graph, our strategy heavily relies on orthogonality. We leave the following as an open problem.

▶ Conjecture 1.2. Let G be a finite graph and let γ be a straight-line embedding of G in the plane. Then, there is a strategy for the human to catch the puppy on γ.

1.2 Related work

A natural variant of Biro's problem, which AEKLMUVV call the *guppy* problem, was introduced by Kouhestani and Rappaport [13]: the guppy is swimming in a simply connected lake, behaving just like the puppy in our problem at hand. The human, however, can only walk along the shore. Kouhestani and Rappaport conjectured that there is always a strategy for the human to catch the guppy. However, this conjectured was settled in the negative when Abel, Akitaya, Demaine, Demaine, Hesterberg, Korman, Ku and Lynch provided a counterexample consisting of an orthogonal polygon where no such strategy exists [1].

As mentioned above, the puppy problem is motivated by the problem of beacon routing. A *beacon* is a point which can be activated to create a magnetic pull. Given a polygonal domain P, the question is now how many beacons need to be placed and activated sequentially to route a point in P to a given goal location. The difference from the puppy problem is thus that beacons cannot move but that several of them can be used. Beacon routing has been studied extensively in the last 10 years, in particular for orthogonal polygonal domains [3, 4, 5, 6, 7, 8, 9, 11, 12, 14, 15, 16].

2 A Strategy for Orthogonal Embeddings

In this section, we prove Theorem 1.1 in two steps. First, in Section 2.1 we show it for the *generic* case, where we assume that no two horizontal line segments of the embedding are on the same height. This restricted setting already highlights the main ideas but is simpler to describe. In Section 2.2 we then adapt our proof to the non-generic case.

2.1 The generic case

We begin by proving the following lemma.

▶ **Lemma 2.1.** *Given a configuration of the human's position h and the puppy's position p and a horizontal line l through h where p is not above l, the puppy will move above l only if the human moves above l.*

Proof. Since all edges are either parallel or orthogonal to l, the only way for the puppy to move above l is to take a vertical edge that crosses l. However, since the puppy is currently below or at the same level as the human, no movement of the human can lead to this: If the human moves horizontally, then taking a vertical edge that crosses l can only increase the distance between the two. Similarly, if the human moves vertically without crossing l, the puppy will only increase its distance by crossing l. ◀

Proof of Theorem 1.1. Let γ be an orthogonal straight-line embedding of a graph G. We describe an algorithm for the human to catch the puppy. The basic idea is that we decrease the size of the part of the embedding containing the puppy in every step by considering parts of the domain as "forbidden." In the following we assume without loss of generality that at the beginning the puppy is not above the human. The algorithm is the following, see Figure 5.

1. Move in the non-forbidden part to the (unique) topmost horizontal edge e. All (vertical) parts of edges that are above this position are now forbidden.
2. If e is a cut edge (i.e., bridge) of the non-forbidden part, move to the connected component of $\gamma \setminus \{e\}$ that contains the location of the puppy and consider the edge (as well as all other components of the graph) forbidden.
3. Otherwise, simply leave the edge and consider it and everything above it forbidden from now on.
4. Repeat this process in the non-forbidden part of the embedding.

We want to prove that whenever we label a new part of the domain as "forbidden," neither the human nor the puppy will ever enter that region again. We make an inductive argument. Our induction hypothesis is that before each iteration, the non-forbidden domain is connected and contains both the human and the puppy. Additionally, the forbidden domain has the property that the puppy will not enter any forbidden part if the human stays in the non-forbidden parts. These invariants are trivially true at the beginning of the process. Now consider the configuration at the beginning of any iteration of the algorithm. By the induction

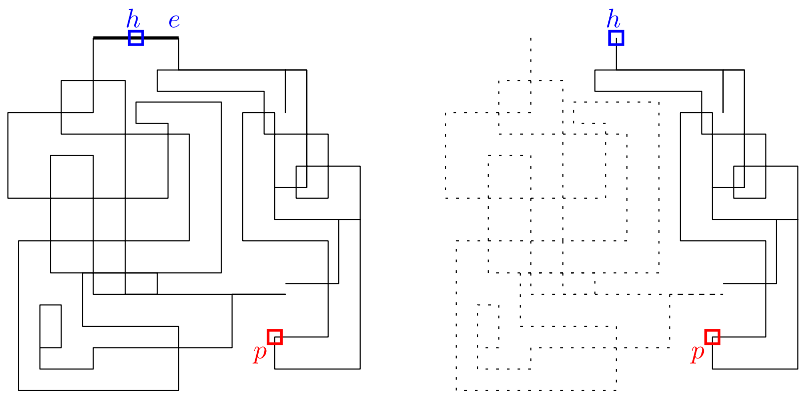


Figure 5 An example of Step 2 of the algorithm.

hypothesis, the non-forbidden domain is connected and thus the human can move to the topmost edge e without entering any forbidden parts. Again, by the induction hypothesis, the puppy is still in the non-forbidden domain when the human reaches e. We argue that the induction hypothesis still holds after cutting off everything above e: by Lemma 2.1, the puppy cannot be above the human as e is the topmost edge. Since we only cut off (parts of) vertical edges, the domain stays connected. Again by Lemma 2.1, the puppy will not enter the newly forbidden domain unless the human does. Now, there are two cases.

1. If e is not a cut edge, then the human simply leaves the edge and considers it forbidden from now on. Now, the non-forbidden domain still contains the human and the puppy and is connected since otherwise e must have been a cut edge. Moreover, since e was the unique topmost horizontal edge and the human will never enter the forbidden parts again, by Lemma 2.1, the puppy cannot enter e either.
2. If e is a cut edge, then the human moves to the connected component that contains the puppy and we forbid e as well as all other connected components. After this, both the human and puppy are still in the non-forbidden domain. By definition, the domain is also still connected. By the same arguments as in the first case, the puppy will stay in the non-forbidden part.

In each step the non-forbidden part of the domain decreases by at least one edge. Since there are only finitely many edges, the human and the puppy will eventually be located on the same edge at which point the human has caught the puppy. ◀

2.2 Allowing horizontal edges of the same height

Now, we are no longer assuming each horizontal edge to be at a different height, meaning that we need to adjust the strategy from the previous section as the topmost horizontal edge is no longer unique. Let m be the height of the topmost horizontal edge in γ, and let $T = \{e_1, \ldots, e_k\}$ be all edges in γ at height m, ordered from left to right. Now, we consider the connected components of $\gamma \setminus T$ (note that this could be a single component). Our goal is to get to a configuration where the human is in the same component as the puppy. Once we are in such a configuration, we can again forbid all edges of T and by Lemma 2.1, the puppy will never leave the component again, unless the human does, meaning that we can recursively continue the strategy on the non-forbidden part.

It thus remains to show that the human can always move into the same component as the puppy. For each component C of $\gamma \setminus T$ denote by $T(C)$ the set of edges in T it is incident to.

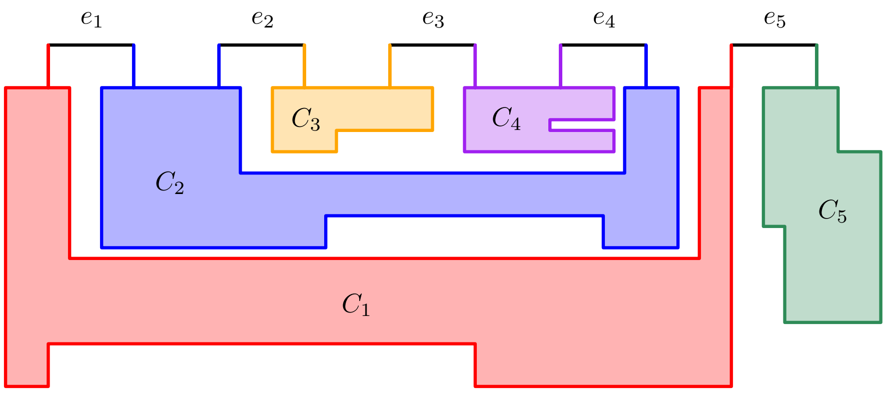


Figure 6 An orthogonal embedding with five components. The components C_1 and C_2 are U-components. The component C_1 dominates C_2, which in turn dominates C_3 and C_4.

We say that a component C of $\gamma \setminus T$ is a *U-component* if there is another component C' that lies entirely inside the region bounded by C and the horizontal line at height m. In this case we say that C *dominates* C'. This gives a nesting of the components, see Figure 6 for an illustration.

▶ **Lemma 2.2.** *Let C_u be a U-component and let $\mathcal{D}$ be the set of all components dominated by C_u (including C_u). If both human and puppy are on components in $\mathcal{D}$ and the human stays on components in $\mathcal{D}$, then the puppy also stays on components in $\mathcal{D}$.*

Proof. Let e^{l} and e^{r} be the leftmost and rightmost edges in $T(C_u)$, respectively. In order to leave $\mathcal{D}$, the puppy must either run leftwards over e^{l} or rightwards over e^{r}. However, the puppy would only do this if the human was on an edge in T left of e^{l} or right of e^{r}, in which case the human is not in $\mathcal{D}$. ◀

Consider now the graph $G_{\mathcal{D}}$ whose vertices are the components that are not dominated by some other component, where two such components are connected whenever they are connected via an edge in T. We claim that $G_{\mathcal{D}}$ is a path: for each vertex C_i of $G_{\mathcal{D}}$ let e_i^{l} and e_i^{r} denote the leftmost and rightmost edge in $T(C_i)$, respectively. The intervals enclosed by e_i^{l} and e_i^{r} must be pairwise disjoint by the above observations. Thus, there is a linear order along which the components are connected, implying that $G_{\mathcal{D}}$ is indeed a path.

The strategy of the human is now the following: walk to the vertex in $G_{\mathcal{D}}$ corresponding to the component C dominating the component where the puppy currently is. As $G_{\mathcal{D}}$ is a path, this can always be achieved. Then, remove all components not dominated by C from consideration. By Lemma 2.2, the puppy will always stay on the remaining graph. If the puppy is also in C, remove everything but C from consideration and recurse. Otherwise, walk into the connected component of $\gamma \setminus C$ in which the puppy is, remove everything else from consideration and recurse. By Lemma 2.2 and the above observations, the puppy will again stay on the remaining graph. As the size of the part of the graph embedding γ under consideration decreases in each recursion step, at some point the human is in the same component C of $\gamma \setminus T$. The human enters this component via an edge in T. Thus, if the puppy is also on an edge in T it must be on the same one and it will run to the human. Otherwise the puppy is below the human at which point we can recurse by considering the topmost edges of C.

References

1 Zachary Abel, Hugo A Akitaya, Erik D Demaine, Martin L Demaine, Adam Hesterberg, Matias Korman, Jason S Ku, and Jayson Lynch. Negative instance for the edge patrolling beacon problem. In *Discrete and Computational Geometry, Graphs, and Games: 21st Japanese Conference, JCDCGGG 2018, Quezon City, Philippines, September 1-3, 2018, Revised Selected Papers 21*, pages 28–35. Springer, 2021.

2 Mikkel Abrahamsen, Jeff Erickson, Irina Kostitsyna, Maarten Löffler, Tillmann Miltzow, Jérôme Urhausen, Jordi Vermeulen, and Giovanni Viglietta. Chasing puppies: Mobile beacon routing on closed curves. *Journal of Computational Geometry*, 13(2):115–150, 2022.

3 Israel Aldana-Galván, Jose Luis Álvarez-Rebollar, Juan C Catana-Salazar, Nestaly Marin-Nevárez, Erick Solís-Villarreal, Jorge Urrutia, and Carlos Velarde. Beacon coverage in orthogonal polyhedra. In *CCCG*, pages 156–161, 2017.

4 Israel Aldana-Galván, Jose Luis Álvarez-Rebollar, Juan Carlos Catana-Salazar, Nestaly Marín-Nevárez, Erick Solís-Villarreal, Jorge Urrutia, and Carlos Velarde. Covering orthotrees with guards and beacons. In *Proceedings of XVII Spanish Meeting on Computational Geometry, Alicante, Spain, July*, pages 26–28, 2017.

5 Sang Won Bae, Chan-Su Shin, and Antoine Vigneron. Improved bounds for beacon-based coverage and routing in simple rectilinear polygons. *arXiv preprint arXiv:1505.05106*, 2015.

6 Sang Won Bae, Chan-Su Shin, and Antoine Vigneron. Tight bounds for beacon-based coverage in simple rectilinear polygons. *Computational Geometry*, 80:40–52, 2019.

7 Michael Biro. *Beacon-based routing and guarding.* PhD thesis, State University of New York at Stony Brook, 2013.

8 Michael Biro, Jie Gao, Justin Iwerks, Irina Kostitsyna, and Joseph SB Mitchell. Combinatorics of beacon-based routing and coverage. In *25th Canadian Conference on Computational Geometry (CCCG 2013)*, pages 1–6, 2013.

9 Michael Biro, Justin Iwerks, Irina Kostitsyna, and Joseph SB Mitchell. Beacon-based algorithms for geometric routing. In *Algorithms and Data Structures: 13th International Symposium, WADS 2013, London, ON, Canada, August 12-14, 2013. Proceedings 13*, pages 158–169. Springer, 2013.

10 Florestan Brunck, Maarten Löffler, and Rodrigo I Silveira. A curve with rotation number one that is not universal for beacon routing. In *Proc. 41st European Workshop on Computational Geometry EuroCG'25*, 2025.

11 Jonas Cleve and Wolfgang Mulzer. Combinatorics of beacon-based routing in three dimensions. *Computational Geometry*, 91:101667, 2020.

12 Bahram Kouhestani. Efficient algorithms for beacon routing in polygons. Master's thesis, Queen's University (Canada), 2016.

13 Bahram Kouhestani and David Rappaport. Edge patrolling beacon. In *20th Japan Conference on Discrete and Computational Geometry, Graphs, and Games (JCDCGGG 2017)*, pages 101–102, 2017.

14 Bahram Kouhestani, David Rappaport, and Kai Salomaa. The length of the beacon attraction trajectory. In *CCCG*, pages 69–74, 2016.

15 Bahram Kouhestani, David Rappaport, and Kai Salomaa. Routing in a polygonal terrain with the shortest beacon watchtower. *Computational Geometry*, 68:34–47, 2018.

16 Thomas C Shermer. A combinatorial bound for beacon-based routing in orthogonal polygons. *arXiv preprint arXiv:1507.03509*, 2015.

On Triangles in Colored Pseudoline Arrangements

Yan Alves Radtke[1], **Balázs Keszegh**[*] [2], **and Robert Lauff**[1]

1 **Institut für Mathematik, Technische Universität Berlin, Germany**
{alves,lauff}@math.tu-berlin.de

2 **HUN-REN Alfréd Rényi Institute of Mathematics and ELTE Eötvös Loránd University, Budapest, Hungary**
keszegh@renyi.hu

Abstract

We consider the faces in pseudoline arrangements in which the pseudolines are colored with two colors. Björner, Las Vergnas, Sturmfels, White, and Ziegler conjecture the existence of a two-colored triangle in such arrangements. We consider variants of this problem. We show that in any non-trivial two-coloring of a pseudoline arrangement there exists a two-colored triangle or quadrangle. We also investigate the existence of a bichromatic triangle assuming certain structures on the coloring.

We turn our attention to the hypergraph whose vertices correspond to the pseudolines of an arrangement and its hyperedges to the triangular faces. Previously, several authors investigated the chromatic number and independence number of hypergraphs whose vertices correspond to the pseudolines of an arrangement and the hyperedges correspond to the faces of any size of the arrangement. We prove that the maximum of the independence numbers of the line-triangle hypergraphs is $n - \Theta(\log n)$.

1 Introduction

An *Euclidean pseudoline arrangement* is a finite collection of bi-infinite, simple curves called *pseudolines* in the Euclidean plane, such that they pairwise cross in exactly one point, which we will call *crossing*. If they exist, we call pseudolines with no crossings on one side *extremal*. An arrangement is *simple*, if no three pseudolines intersect in a common point. We only consider simple arrangements.

A pseudoline arrangement gives rise to a collection of *vertices*, *edges* and *faces*, see Figure 1. We call a bounded face a *triangle* resp. *quadrangle*, if it is supported by exactly three resp. four pseudolines. Sometimes we consider the area between three pseudolines and call it a *non-empty triangle*.

The minimum and maximum number of triangles in pseudoline arrangements are known[8, 6]. Other questions about triangles remain open. We will consider arrangements whose pseudolines are colored blue and red so that at least one pseudoline of each color exists. We call such arrangements *bicolored*. In 1993, Björner, Las Vergnas, Sturmfels, White, and Ziegler asked about the existence of bichromatic triangles in bicolored arrangements.

▶ **Conjecture 1.1** ([5, p. 280])**.** *Every bicolored arrangement has a bichromatic triangle.*

Another way to look at this problem is to consider the *triangle-pseudoline incidence graph*. As its vertices, take the triangles and pseudolines and add an edge between a pseudoline and a triangle if the pseudoline supports the triangle. It is intuitive to ask about the connectivity of this graph.

[*] Research supported by the National Research, Development and Innovation Office – NKFIH under the grant K 132696 and by the ERC Advanced Grant "ERMiD". This research has been implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the ELTE TKP 2021-NKTA-62 funding scheme.

▶ **Conjecture 1.2** ([5, p. 278]). *The triangle-pseudoline incidence graph of a pseudoline arrangement is connected.* [1]

Note that Conjecture 1.1 and Conjecture 1.2 are equivalent. The existence of bichromatic triangles in a bicolored straight line arrangement has a simple proof: Shift the blue sub-arrangement up and down. Consider the first moment the combinatorics of the arrangement changes. At this point, a blue-blue crossing moved over a red line or a red-red crossing moved over a blue line. The crossing and the line form a bichromatic triangle in the original arrangement.

A generalisation of this result was given in [9]: An arrangement $\mathcal{A} = \{\ell_1, \dots, \ell_n\}$ is *approaching*, if every pseudoline ℓ_i is the graph of a function f_i, such that $f_i - f_j$ is strictly monotone increasing, for $i < j$. These pseudolines can be shifted and the collection of curves remains an arrangements of pseudolines. This implies that Conjecture 1.1 holds for all approaching arrangements.

The number of approaching arrangements has asymptotics similar to the class of all arrangements, but examples of small arrangements that are not isomorphic to an approaching arrangement are known.

Although the proof in the case of approaching arrangements is very elegant and intuitive, not much progress has been made towards the general case so far. We present results about easier variants of the question.

Also towards the goal of understanding triangles in pseudoline arrangements, we consider the *line-triangle hypergraph*, which is the 3-uniform hypergraph whose vertices correspond to the pseudolines and a triple forms a hyperedge if the corresponding lines form a triangle. Then Conjecture 1.1 claims that every true 2-coloring of this hypergraph has a non-monochromatic hyperedge. We present a theorem about the independence number of this hypergraph, i.e. we consider how large the discrepancy of the cardinalities of the two color classes in an arrangement can be while not including any monochromatic triangles.

We will consider Euclidean arrangements as *marked arrangements*, which have a fixed unbounded face, which we will call the *north face*. This gives use a canonical numbering of the pseudolines, after choosing the north face, see Figure 1.This induces an orientation on every triples of lines: we say that a triple of lines has "-" orientation, if the middle line goes above the crossing of the other two, otherwise it is $+$, see Figure 1.

▶ **Definition 1.3** ([10]). The function $\sigma : \binom{[n]}{k} \to \{-,+\}$ is a rank k signotope on n elements, if for all

$$S \coloneqq \{x_1 < x_2 < \cdots < x_{k+1}\} \subseteq [n],$$

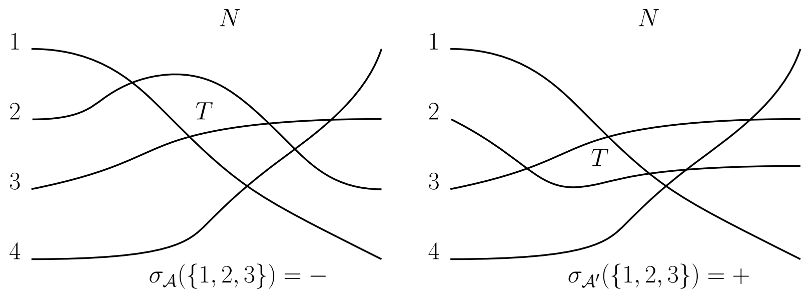
the sequence

$$\sigma(S \setminus x_1), \sigma(S \setminus x_2), \dots, \sigma(S \setminus x_{k+1})$$

has at most one sign change.

The orientation of triples of pseudolines of $\mathcal{A}$ give rise to the rank 3 signotope $\sigma_{\mathcal{A}}$. Conversely, every rank 3 signotope can be seen as triangle orientations of a pseudoline arrangement [10]. It is possible to locally mutate a pseudoline arrangement $\mathcal{A}$ by *flipping* a triangle T to obtain $\mathcal{A}'$, i.e. changing the orientation of the pseudolines supporting T. This is equivalent to $\sigma_{\mathcal{A}}$ and $\sigma_{\mathcal{A}'}$ differing on a single value, corresponding to T. See Figure 1. The most elementary tool to locate triangles is the so called *sweeping lemma for pseudolines*.

[1] The original (weaker) conjecture regarded projective arrangements.



N
1
2
3
4
T
$\sigma_{\mathcal{A}}(\{1,2,3\}) = -$
N
1
2
3
4
T
$\sigma_{\mathcal{A}'}(\{1,2,3\}) = +$

Figure 1 The north face is designated by N. The face T is a triangle. The crossings (1,2) and (1,3) are vertices, which are connected by an edge. The left arrangement corresponds to the all "$-$" signotope. The right arrangement is obtained by flipping T.

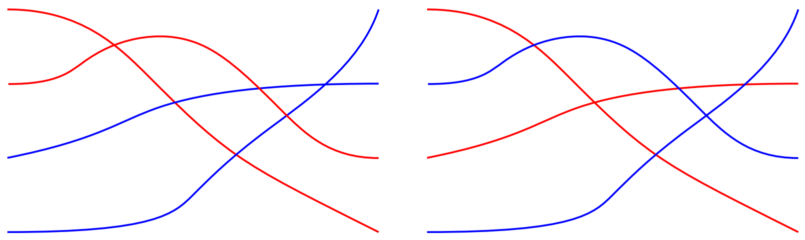


Figure 2 The arrangement on the left is block-bicolored. The arrangement on the right is not.

▶ **Lemma 1.4** ([10]). *Let ℓ be a pseudoline in a marked arrangement. If there is a crossing above ℓ, then there is a crossing above ℓ that forms a triangle supported by ℓ. The same holds for crossings below ℓ.*

There are two a-priori different orders on the set of signotopes with the same parameters: the *inclusion order*, where $\sigma_1 \leq \sigma_2$, when $\sigma_1^{-1}\{+\} \subseteq \sigma_2^{-1}\{+\}$ and the *single-step inclusion order*, where $\sigma_1 \leq \sigma_2$, if there is a sequence of $-$ to $+$ flips that transform σ_1 into σ_2. The following was proven by Felsner and Weil.

▶ **Theorem 1.5** ([11]). *The single-step inclusion order and the inclusion order on rank three signotopes coincide.*

1.1 Our results about bichromatic faces

We consider multiple weaker variants of Conjecture 1.1. We first assume certain structures on the coloring of the arrangements. The following is already implicit in [3].

▶ **Theorem 1.6.** *A bicolored arrangement with at most 5 red pseudolines has a bichromatic triangle.*

This implies that if we color an arrangement with $n/5$ colors and use all of them, then there is a non-monochromatic triangle. It also implies the conjecture holds for all arrangements with $n \leq 11$. We call an (unmarked) bicolored pseudoline arrangment *block-bicolored*, if we can choose a north cell, such that the first pseudolines in the numbering induced by the north cell are red and the rest are blue, see Figure 2. The following result was first proven by Felsner [personal communication]. Here we provide a simpler proof.

▶ **Theorem 1.7** ([2]). *If a pseudoline arrangement is block-bicolored it has a bichromatic triangle.*

We then broadened the question and considered the existence of bichromatic faces with low complexity.

▶ **Theorem 1.8.** *Every bicolored arrangement contains a bichromatic triangle or a bichromatic quadrangle.*

1.2 The line-triangle hypergraph

Let $H_{face}(L)$ denote the line-face hypergraph of the arrangement defined by a set of lines L, i.e., the hypergraph whose vertices correspond to the lines and a subset forms a hyperedge if the corresponding lines form a face in the arrangement (including the unbounded faces). Properties of this hypergraph were regarded earlier by Bose et al.[7] and then their initial results were improved by Ackerman et al. [1] and Balogh and Solymosi [4].

Let α be the independence number of a hypergraph, that is, the largest size of an independent set. The following theorem summarizes previous knowledge about the relevant parameters of line-face hypergraphs:

▶ **Theorem 1.9** ([7, 1, 4])**.**
- $\Omega(\sqrt{n \log n}) = \min\limits_{|L|=n} \alpha(H_{face}(L)) \leq n^{5/6+o(1)}$,
- $n/2 \leq \max\limits_{|L|=n} \alpha(H_{face}(L)) < \frac{2}{3}n$,
- $n^{1/6-o(1)} \leq \max\limits_{|L|=n} \chi(H_{face}(L)) = O(\sqrt{n/\log n})$.

We note that the proofs of Theorem 1.9 work also if we replace lines by pseudolines. From now on, we only consider pseudoline arrangements. Also, the first and the third result hold if we consider the sub-hypergraph containing only the bounded faces as hyperedges.

Instead of the line-face hypergraph, we can consider the line-triangle hypergraph, which we denote by $H_\Delta(L)$. The next is a direct corollary of Theorem 1.9, using that the line-triangle hypergraph is a subhypergraph of the line-face hypergraph. We also need that in the construction in [4] it is enough to consider only triangular faces to obtain a small independence number.

▶ Corollary 1.
- $\Omega(\sqrt{n \log n}) = \min\limits_{|L|=n} \alpha(H_\Delta(L)) \leq n^{5/6+o(1)}$,
- $n^{1/6-o(1)} \leq \max\limits_{|L|=n} \chi(H_\Delta(L)) = O(\sqrt{n/\log n})$.

Our contribution in this area is the following theorem.

▶ **Theorem 1.10.** *The maximum of $\alpha(H_\Delta(\mathcal{A}))$ over every arrangement of pseudolines $\mathcal{A}$ of size n is* $\max\limits_{|\mathcal{A}|=n} \alpha(H_\Delta(\mathcal{A})) = n - \Theta(\log n)$.

This implies the same upper bound for families of lines. We do not provide a proof that our construction works with straight line arrangement.

2 Proof sketches

We present proof sketches for our main results. See the soon to appear full-version for detailed proofs.

Proof Sketch of Theorem 1.6. If the red sub-arrangment has an extremal line, either a simple sweeping argument shows the existence of a bichromatic triangle or we can delete the extremal line and continue with a smaller red sub-arrangement. The only arrangement without extremal line with $n \leq 5$ is the 5-star, where an explicit argument can be given. ◀

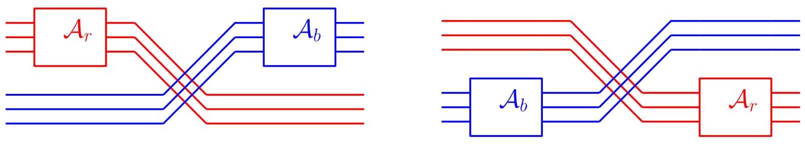


Figure 3 Both extremal arrangements with prescribed sub-arrangements

Proof of Theorem 1.7. Given a block-bicolored arrangement $\mathcal{A}$, we call the arrangement induced by the red pseudolines $\mathcal{A}_r$ and the arrangement induced by the blue pseudolines $\mathcal{A}_b$. Consider the arrangements shown in Figure 3. In the arrangement to the left, call it $\mathcal{A}_{min}$, for every triple of lines, not all of them of the same color, the pseudoline with the highest index is going below the crossing of the other two. This implies that every bichromatic triple has $-$ orientation. A similar argument implies that every bichromatic triple in the right arrangement, call it $\mathcal{A}_{max}$, has orientation $+$. The remaining, monochromatic, triples in $\mathcal{A}$, $\mathcal{A}_{max}$ and $\mathcal{A}_{min}$ have the same orientation. By the observation above clearly $\mathcal{A}_{min} \leq \mathcal{A} \leq \mathcal{A}_{max}$ holds in the inclusion order. By Theorem 1.5, there is a sequence (possibly of length zero!) of "$-$ to $+$" triangle flips, transforming $\mathcal{A}$ into $\mathcal{A}_{max}$. Since both arrangements agree on the monochromatic triples, the sequence only consists of bichromatic triangle flips. A similar statement holds for $\mathcal{A}_{min}$. Clearly $\mathcal{A}_{min} \neq \mathcal{A}_{max}$, so not both $\mathcal{A} = \mathcal{A}_{min}$ and $\mathcal{A} = \mathcal{A}_{max}$ can hold, so not both flip sequences can be empty, so there exists at least one bichromatic triangle in $\mathcal{A}$. Moreover if both $\mathcal{A} \neq \mathcal{A}_{min}$ and $\mathcal{A} \neq \mathcal{A}_{max}$, $\mathcal{A}$ has two bichromatic triangles, of opposite orientation. ◀

Note that this proof gives us a sequence of arrangements which look like we shift a sub-arrangement to the left, similar to the proof for line arrangements.

We will use a sweeping lemma for lenses, which was stated in [3].

▶ **Lemma 2.1** (Lens Sweeping Lemma). *Let Q be a lens bounded by two curves L and R. Assume there is a collection of curves inside Q which pairwise intersect at most once and where every curve intersects L and R exactly once. If there is a crossing inside Q, there is a triangle that is supported by L.*

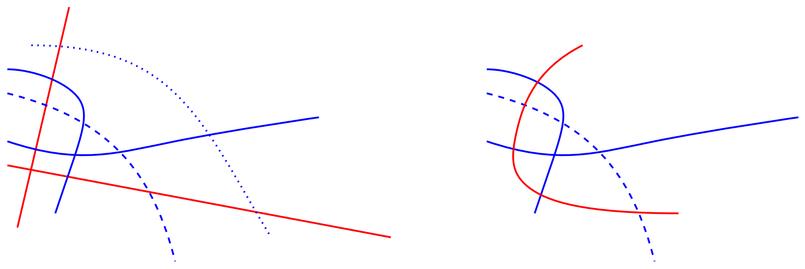


Figure 4 Illustration for the proof of Theorem 1.8. The dashed blue and both red lines form a lens.

Proof sketch of Theorem 1.8. In a bicolored arrangement $\mathcal{A}$, we consider the sub-arrangement induced by the red pseudolines and a single blue pseudoline ℓ_a. There is a blue-red-red triangle T in this subarrangement supported by the blue pseudoline, by Lemma 1.4. Call the red lines ℓ_b and ℓ_c, respectively.

We will only consider the parts of the arrangement $\mathcal{A}$ inside the non-empty triangle T. By infinite descent, we can assume that all curves in T intersect ℓ_a.

We then join ℓ_b and ℓ_c to a single curve ℓ_d and consider the lens formed by ℓ_a and ℓ_d, see Figure 4. By a short case distinction and by using Lemma 2.1 we find a bichromatic triangle in the lens. After reintroducing the red-red crossing this triangle possibly becomes an empty bichromatic quadrangle.

◀

▶ Remark. In the case where we find a quadrangle, this quadrangle is red-red-blue-blue.

Proof Sketch of Theorem 1.10. We first consider the red sub-arrangement. By the Erdős-Szekeres theorem applied in the dual setting to pseudolines, there we find a cyclic sub-arrangement of size $\Theta(\log(n))$. The cyclic arrangement on k pseudolines has $k-2$ triangles. Any non-empty triangle in an arrangement contains a triangle. Since each pseudoline added to the cyclic arrangement can intersect at most two of its triangles, we need to add at least $\Theta(\log(n))$ blue pseudolines to destroy all the red triangles. This yields the upper bound.

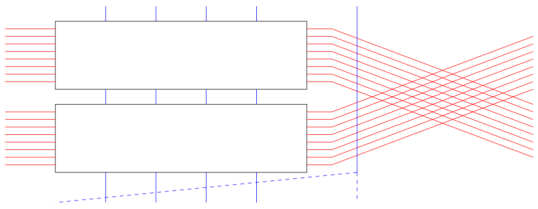


Figure 5 The bottom block has no blue crossings and the top block either has all blue crossings or no blue crossings. The extra blue line either crosses all other blue lines below, or it crosses none.

For the lower bound we inductively construct an arrangement with 2^n red pseudolines and n blue pseudolines without monochromatic triangles, see Fig 5. ◀

3 Discussion

The conjecture about the existence of a bichromatic triangle is solved if there are at most 5 red pseudolines. One particular case of 6 red pseudolines which we could not solve in general (that is, for any addition of blue pseudolines) and find especially interesting is depicted in Figure 6. Note that in any extension with blue pseudolines, as every red pseudoline must have an incident triangle on both sides, either one of them is bichromatic and we are done, or no gray triangles on the figure can be crossed by a blue pseudoline.

The bounds regarding the face-hypergraph and triangle-hypergraph still have large gaps except for the one case in Theorem 1.10. We find the gap between $n/2$ and $2n/3$ for $\max_{|L|=n} \alpha(H_{face}(L))$ a particularly interesting problem.

As an approach to proving Theorem 1.8 we attempted to show the following result which we conjecture to be true.

▶ **Conjecture 3.1.** *Given a blue k-gon P with at least $k-4$ red pseudolines passing through, we find a bichromatic triangle or quadrangle on its inside boundary.*

It is easy to see that $k-4$ is tight if there are no crossings between the red pseudolines.

Acknowledgement

We thank the organizers of the Order and Geometry Workshop 2024 and Stefan Felsner for helpful discussions.

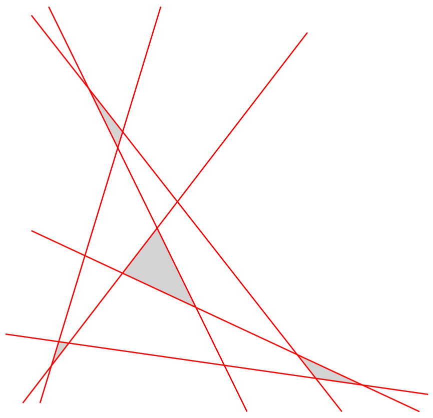


Figure 6 Six red pseudolines to which we need to add several blue pseudolines.

References

1 Eyal Ackerman, János Pach, Rom Pinchasi, Radoš Radoičić, and Géza Tóth. A note on coloring line arrangements. *The Electronic Journal of Combinatorics*, 21(2), 2014.

2 Yan Alves Radtke. *Signotopes: Bichromatic Triangles, Pseudoconfigurations and Extensions.* Master's thesis, Technische Universität Berlin, 2024.

3 Yan Alves Radtke, Stefan Felsner, Johannes Obenaus, Sandro Roch, Manfred Scheucher, and Birgit Vogtenhuber. Flip graph connectivity for arrangements of pseudolines and pseudocircles. *Proceedings SODA 24*, 2024.

4 Jozsef Balogh and Jozsef Solymosi. On the number of points in general position in the plane. *Discrete Analysis*, 2018.

5 Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Günter Ziegler. *Oriented Matroids*, volume 46 of *Encyclopedia of Mathematics*. Cambridge University Press, Cambridge, 1993.

6 Jérémy Blanc. The best polynomial bounds for the number of triangles in a simple arrangement of n pseudo-lines. *Geombinatorics*, 21:5–17, 2011.

7 Prosenjit Bose, Jean Cardinal, Sébastien Collette, Ferran Hurtado, Matias Korman, Stefan Langerman, and Perouz Taslakian. Coloring and guarding arrangements. *Discret. Math. Theor. Comput. Sci.*, 15(3):139–154, 2013.

8 Stefan Felsner and Klaus Kriegel. Triangles in euclidean arrangements. *Discrete & Computational Geometry*, 22:429—-438, 1999.

9 Stefan Felsner, Alexander Pilz, and Patrick Schnider. Arrangements of approaching pseudo-lines. *Discrete & Computational Geometry*, 67:380–402, 2022.

10 Stefan Felsner and H. Weil. Sweeps, arrangements and signotopes. *Discrete Applied Mathematics*, 109(1–2):67–94, 2001.

11 Stefan Felsner and Helmut Weil. A theorem on higher bruhat orders. *Discrete & Computational Geometry*, 23(1–2):121–127, 2000.

On Planar Unit-Length Linear Linkages in Polygonal Domains[*]

Thomas Depian[1], Carolina Haase[2], Martin Nöllenburg[1], and André Schulz[3]

1 **Algorithms and Complexity Group, TU Wien, Austria**
`{tdepian, noellenburg}@ac.tuwien.ac.at`

2 **Trier University, Germany**
`haasec@uni-trier.de`

3 **FernUniversität in Hagen, Germany**
`andre.schulz@fernuni-hagen.de`

Abstract

A linkage $\mathcal{L}$ is given by a graph G and a function ℓ specifying the length of the edges in G. It is $\exists\mathbb{R}$-complete to decide if $\mathcal{L}$ has a planar embedding with straight-line edges [Abel et al., SoCG'16]. The reduction uses unit-length edges, but G needs to have 3-connected blocks. Here, we consider the problem of finding a planar embedding of $\mathcal{L}$ inside a polygonal domain P when G is a path with prescribed start and end point and $\ell \equiv 1$. Despite the restricted setup, we show NP-hardness for the general setting and provide an efficient algorithm if G has three edges and P is convex.

1 Introduction

Every planar graph has a planar embedding with straight-line edges [10, 21], which can be computed in linear time [8, 19]. However, in many applications, e.g., protein folding [13], motion planning [14], and cartograms [4, 5], we are not interested in just some (planar) straight-line drawing of G but require that each edge e of G has a predetermined length $\ell(e)$.

The study of edge-length constrained graphs, also known as *linkages*, is old and famous results such as Kempe's universality theorem [16, 17] date back to the 1870s. Planarity is not always required, depending on the application. By now, linkages and their drawings, called *configurations*, have been well-studied [1, 11]. The book by Demaine and O'Rourke [9] lists many results and applications. Finding a planar configuration for a linkage is $\exists\mathbb{R}$-complete [1], even if it is *unit-length*, that is all edge lengths are the same. A linkage whose graph is a path, also called a *linear* linkage, can be trivially realized. However, if the linear linkage has to be realized inside a polygonal domain with given endpoints, the complexity is unknown. We remark that without specified endpoints, realizing a linear linkage inside a polygonal domains boils down to finding a placement for (one of) its longest edge(s). If it exists, we can draw the remaining edges arbitrarily close next to the longest one.

Apart from *realizing* a linkage, the problem of deciding whether two configurations can be *reconfigured* into one another by a continuous, planar, and edge-length preserving motion is well-studied [2, 3, 6, 15, 22]. Answering this reconfiguration question is PSPACE-hard even if we relax the planarity requirement or if the linkage is linear and the configuration must avoid obstacles in the plane [11, 14]. The problem remains NP-hard for obstacles made up of

[*] TD and MN acknowledge support from the Vienna Science and Technology Fund (WWTF) [10.47379/ICT22029]. This work started at the 19th European Research Week on Geometric Graphs (GGWeek) in Trier.

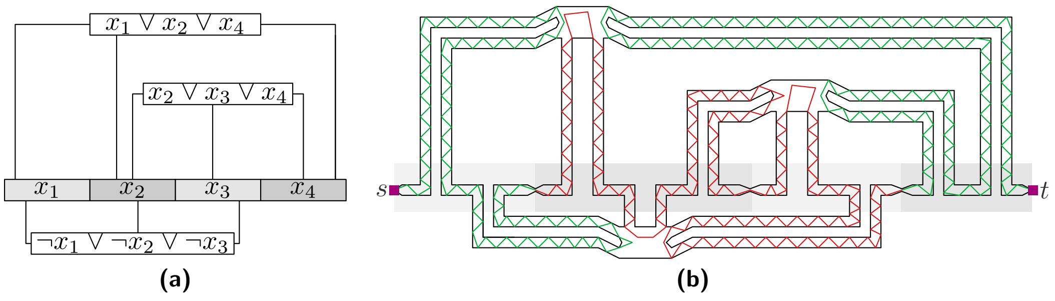


Figure 1 A **(a)** formula φ and the **(b)** schematization of the constructed instance of ULLLR.

four axis-aligned segments [12]. However, the above reductions require non-unit edge lengths. The flat-foldability of linear unit-length linkages has been considered inside equilateral triangles [20]. The complexity of the reconfiguration problem for linear linkages within a polygonal domain remains open, although any two of its planar configurations can be transformed into each other in the absence of a polygonal domain [6].

Problem Definition. Let $G = (V, E)$ be a simple undirected graph on n vertices $V = \{v_1, \ldots, v_n\}$ and m edges E and let $\ell \colon E \to \mathbb{R}^+$ be a function assigning each edge $e \in E$ a *length* $\ell(e) > 0$. A *configuration* Γ of a linkage $\mathcal{L} = (G, \ell)$ is a straight-line drawing of G such that we have $\|\Gamma(u) - \Gamma(v)\|_2 = \ell(e)$ for any $e = uv \in E$. A configuration Γ is *planar* if no two edges cross and it *lives* inside a given polygonal domain $P \subset \mathbb{R}^2$, i.e., a closed, multiply connected region of $\mathbb{R}^2$ [18], if $\Gamma \subset P$. In this abstract, we consider the following problem.

▶ **Problem** (ULL Linkage Realizability (ULLLR))**.** We are given a unit-length linear linkage $\mathcal{L} = (G, \ell)$, a polygonal domain $P \subset \mathbb{R}^2$, and two points $s, t \in P$. Does there exist a planar configuration Γ of $\mathcal{L}$ that lives inside P such that $\Gamma(v_1) = s$ and $\Gamma(v_n) = t$?

Our main contribution is establishing NP-hardness of the above-introduced problem:

▶ **Theorem 1.** ULL Linkage Realizability *is* NP*-hard.*

Due to space constraints, technical details are deferred to an upcoming full version.

2 Overview of the Hardness Reduction

We give a reduction from the NP-complete problem Planar Monotone 3-Sat [7]. An instance $\varphi = (\mathcal{X}, \mathcal{C})$ of this problem consists of variables $\mathcal{X} = \{x_1, \ldots, x_N\}$ and clauses $\mathcal{C} = \{c_1, \ldots, c_M\}$ partitioned into the positive clauses $\mathcal{C}^+$ containing only non-negated literals and negative clauses $\mathcal{C}^-$ containing only negated literals. The clause-variable incidence graph has to have an embedding where all vertices for variables are on a horizontal line, which separates the vertices for $\mathcal{C}^+$ and $\mathcal{C}^-$. Note that we can compute a planar rectilinear drawing $\mathcal{E}$ of the incidence graph with polynomial coordinates; see Figure 1a for an example [4, 5, 7].

In our reduction, we create a unit-length linear linkage $\mathcal{L} = (G, \ell)$ of size n, where we specify n in the end. Conceptually, our construction will force a configuration Γ of $\mathcal{L}$ to visit each variable of φ and each clause it is contained in, set truth assignments for the former and verify their truth status for the latter components. The path that Γ follows will closely resemble the drawing $\mathcal{E}$ of φ, see also Figure 1b, and we will replace each edge and vertex of $\mathcal{E}$ with a gadget, which is a part of the polygonal domain P. We highlight in each gadget dedicated regions in the plane, in the following called *areas*, where Γ must pass through and we create P by gluing the individual gadgets together at these areas.

3 Gadgets of the Hardness Reduction

We now describe on a high level each of the gadgets. Note that all gadgets are agnostic to translations and rotations in the plane. As indicated above, every gadget F contains pairs of entry and exit areas: Once the configuration enters a gadget through an entry area, the construction ensures that it must leave the gadget at the respective exit area. They are specified as quadrants of discs of radius ε, where $\varepsilon < 0.1$ is a small constant whose value we specify in the end, and denoted as $^{\rightarrow}◣_{i,t}(F)$ and $◣^{\rightarrow}_{i,t}(F)$ for a gadget F related to a possible truth assignment $t \in \{0,1\}$ to the variable x_i, i.e., $x_i = t$, respectively. Furthermore, we consider for each *entrance and exit* area the triangle that is inscribed in the same quadrant of a disc of radius $\varepsilon/2$, which we call the *start* and *end* area of a gadget, respectively. With our construction, we ensure that for every point p in the start area, there exists a configuration that starts at p and places a vertex somewhere in the respective end area.

Edge Gadget. There are three types of edge gadgets: tunnels, bends, and shifters. *Tunnels* will inhabit sawtooth-like shaped pairs of unit-segments of height 0.6 and width 1.6. Any configuration Γ can embed at most two edges inside a tunnel enforced by the *obstacles*, i.e., holes in the polygonal domain, of the tunnel depicted in Figure 2a. The edges *zig-zag* around the obstacles by alternating the placement of the vertices between a placement at (or near) the top and the bottom of the tunnel. This allows us to define two equivalence classes on the configurations depending on the side of the tunnel where they place the first vertex. They correspond to the truth assignment to a variable $x_i \in \mathcal{X}$, and we color areas and configurations from the classes in red (🟥) and green (🟩) in the figures, depending whether they correspond to $x_i = 0$ or $x_i = 1$, respectively. We place the first pair at the lower side of the tunnel and the second pair at the upper side of the tunnel as indicated in Figure 2a.

Tunnels are accompanied by *bends*, which force the configuration Γ to perform a $90°$ turn and are depicted in Figure 2b. Note that the obstacles force the green configuration to draw one edge (almost) horizontal and one (almost) vertical. Thus it performs, compared to the red configuration, a small detour to ensure that both configurations place the same number of vertices inside the gadget. This is crucial to ensure correctness of the reduction and is the main difficulty in constructing the gadget. Observe that the bend has at its start and end a height (or width) of 0.6, allowing us to attach tunnels on either of its ends. The entrance and exit areas of a bend are analogous to those of a tunnel.

Tunnels and bends can only start and end at specific coordinates due to their construction. With a *shifter*, we can shift tunnels up and down by 0.2 to give us more flexibility in the construction of the clause gadget. Observe in Figure 2a that inside a tunnel the distance of the endpoints of an edge is approximately 0.8 and 0.6 in x- and y-direction, respectively. With the gadget from Figure 2c, we can force an inverted behavior to move a configuration over the course of two edges up (or down) by 0.2. The shifter consists of the main part from Figure 2c on whose two sides we attach a tunnel to help us show its properties.

Clause Gadget. The main part of the clause gadget for the clause $x_i \vee x_j \vee x_k$ is depicted in Figure 3 and has multiple obstacles that leave only seven narrow (possibly intersecting) *corridors* inside the gadget to limit how a configuration Γ can interact with it. In our reduction, we force the configuration to enter the main part three times, first via the entrance

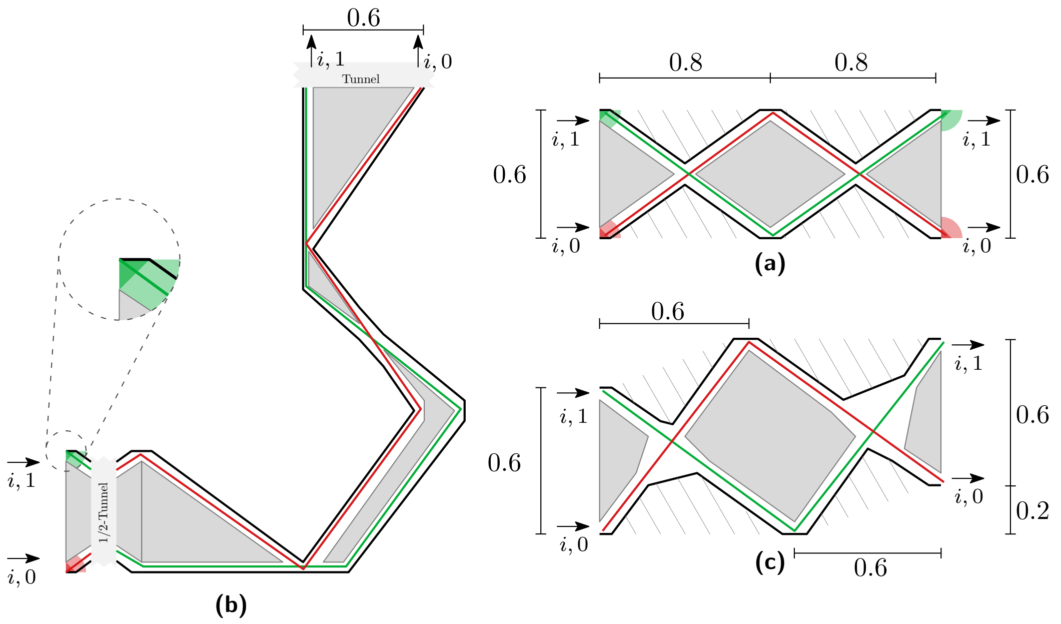


Figure 2 A **(a)** tunnel, **(b)** bend, **(c)** and the main part of a shifter with configurations through them. We hatch the outside of the polygonal domain if there is risk of confusion.

$^{\rightarrow}\blacktriangle_{i,t_i}$, then via $^{\rightarrow}\blacktriangle_{j,t_j}$, and finally via $^{\rightarrow}\blacktriangle_{k,t_k}$, for $t_i, t_j, t_k \in \{0,1\}$. Observe that the distance between $^{\rightarrow}\blacktriangle_{i,t_i}$ and $\blacktriangle^{\rightarrow}_{i,t_i}$ can be spanned by a linkage of length two. The corridors leave little choice for Γ: If Γ enters the main part via $^{\rightarrow}\blacktriangle_{i,0}$, it is forced to leave it via $\blacktriangle^{\rightarrow}_{i,0}$, otherwise, i.e., if it enters the main part via $^{\rightarrow}\blacktriangle_{i,1}$, it is forced to leave it via $\blacktriangle^{\rightarrow}_{i,1}$. Note that placing a vertex in an area for x_j or x_k is impossible due to the unit-length requirement of the edges paired with the corridors. The same holds true for the entrance and exit areas corresponding to x_k. The three corridors in the middle constrain how a configuration can reach $\blacktriangle^{\rightarrow}_{j,t_j}$ from $^{\rightarrow}\blacktriangle_{j,t_j}$ using three edges. In particular, if Γ enters the main part via $^{\rightarrow}\blacktriangle_{j,0}$, the gadget contains two corridors, effectively giving the configuration the flexibility to lean more towards the left or right side of the main part; compare also Figure 3b and Figure 3c. Conversely, i.e., if Γ enters via $^{\rightarrow}\blacktriangle_{j,1}$, there is again only one corridor, giving the configuration little freedom in placing the remaining vertices.

The construction allows for the following crucial observation; compare also Figures 3b to 3e: If a configuration Γ enters the main part via $^{\rightarrow}\blacktriangle_{i,0}$ *and* $^{\rightarrow}\blacktriangle_{k,0}$, a planar configuration that enters the main part via $^{\rightarrow}\blacktriangle_{j,0}$ becomes impossible. On the other hand, if Γ enters the main part via $^{\rightarrow}\blacktriangle_{i,1}$ *or* $^{\rightarrow}\blacktriangle_{k,1}$, it uses a corridor that does not intersect with the ones for $^{\rightarrow}\blacktriangle_{j,0}$, allowing a planar configuration even if Γ enters the main part via $^{\rightarrow}\blacktriangle_{j,0}$. The corridor connecting $^{\rightarrow}\blacktriangle_{j,1}$ with $\blacktriangle^{\rightarrow}_{j,1}$ can always be used.

Finally, we remark that for a suitable small constant ε it is not possible to enter the clause gadget at some entrance area assigned for one variable and leave it at an entrance/exit area assigned to a different variable. To see this recall that we have seven possible "routes" in which the linkage is intended to pass through the gadget (two for x_i/x_k and three for x_j); compare this also to the seven corridors. We can find a constant α such that for every route all possible actual configurations are contained in a polygonal corridor of width at most $\alpha\varepsilon$, which we use to refine the original corridors. The obstacles of P in the clause gadget are then defined by the points outside of the refined corridors. Let γ be the smallest "turning

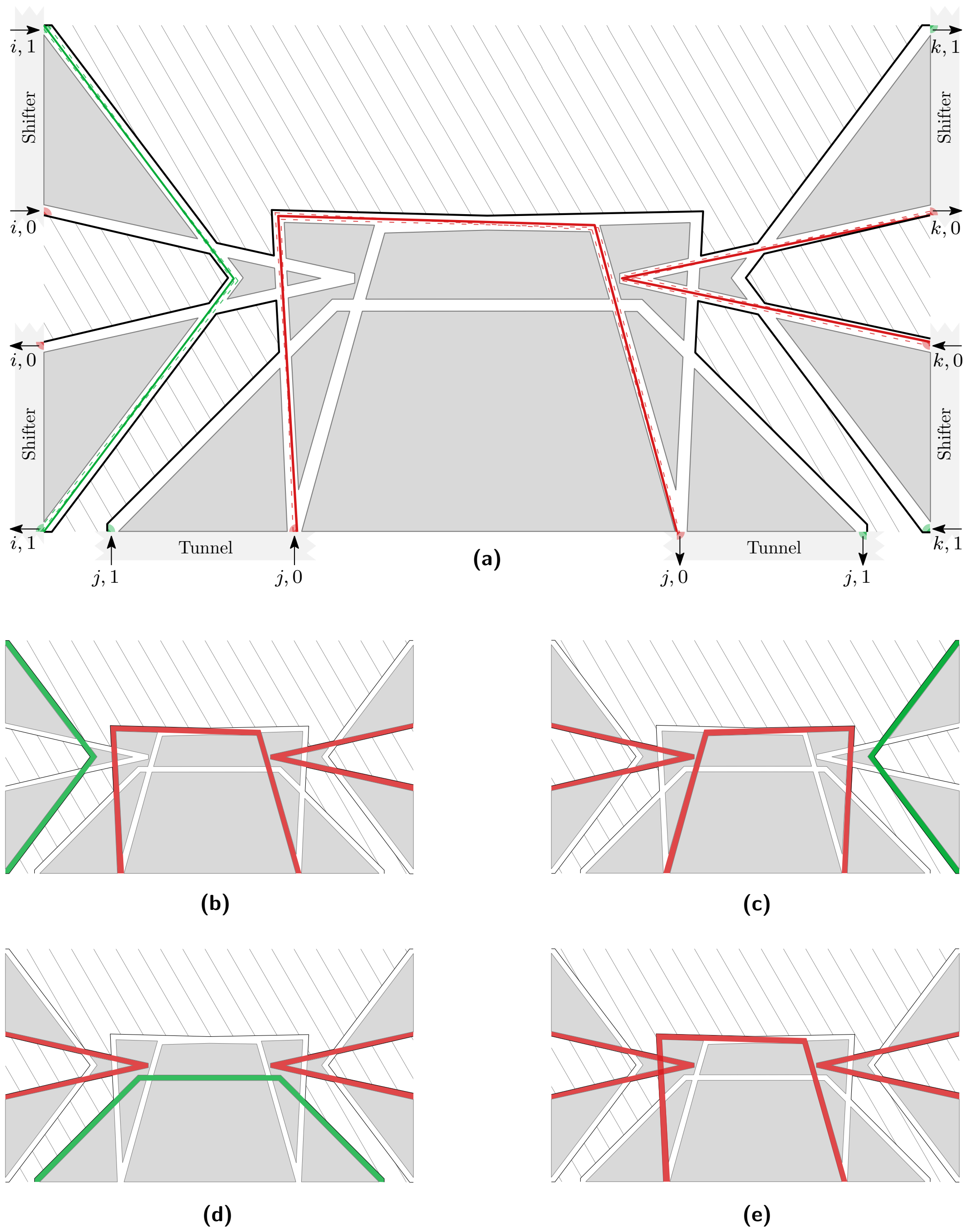


Figure 3 **(a)** The main part of the clause gadget and **(b)**–**(e)** different configurations Γ through it. Dashed lines indicate different possibilities for the configuration to pass through the corridors.

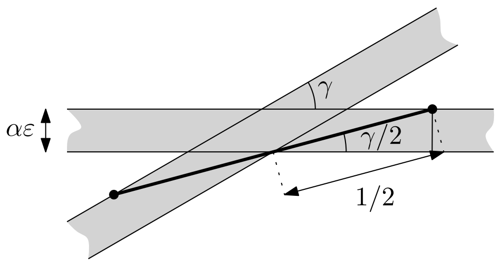


Figure 4 A segment of length 1 can only have endpoints in two corridors if $\gamma \leq 2 \arcsin 2\alpha\varepsilon$.

angle" for two intersecting corridors. Note that γ is independent from $\alpha\varepsilon$. In order to pass from one corridor to another, there has to be a segment of length one with endpoints in distinct corridors. A simple calculation shows that this is only possible if $\gamma \leq 2 \arcsin 2\alpha\varepsilon$; see Figure 4. Thus, picking $\varepsilon \leq \frac{\alpha}{2} \sin \frac{\gamma}{2}$ ensures that we cannot deviate from the intended route through the clause gadget. Observation 1 summarizes this.

▶ **Observation 1.** *Let Γ be a planar configuration of $\mathcal{L}$ that enters the main part C' of a clause gadget three times. For any variable x_i, truth assignment $t \in \{0,1\}$ to x_i, and vertex $v \in V$, we have that $\Gamma(v) \in {}^{\rightarrow}◣_{i,t}(C')$ implies that there is some $v' \in V$ with $\Gamma(v') \in ◣_{i,t}^{\rightarrow}(C')$. Furthermore, there is some $v'' \in V$ such that $\Gamma(v'') \in {}^{\rightarrow}◣_{z,1}(C')$ for some $z \in \{i,j,k\}$.*

We attach on the left and right side of the main part two shifters and at the bottom side two tunnels each to obtain the clause gadget and unify the entrance and exit areas.

Variable Gadget. The variable gadget for a variable $x_i \in \mathcal{X}$ consists of three main components with different roles: making Γ "set" the truth assignment $x_i = t$, propagating this to all relevant clauses and "resetting" Γ before entering the variable gadget of x_{i+1}, if it exists. The first component of the variable gadget is depicted in Figure 5a. It consists of a triangular obstacle, forcing the configuration to place the next vertex either at the top or bottom end of the gadget, corresponding to setting the variable to *true* or *false*, respectively. The base of the triangular obstacle has a height of 0.6, allowing us to attach a tunnel. However, a naïve construction of the triangle would require to place its tip at an irrational coordinate. By reducing the height of the triangle slightly, we avoid this and force the linkage to place a vertex inside $◣_{i,t}^{\rightarrow}$ for $t \in \{0,1\}$, effectively setting $x_i = t$. The third component of the variable gadget, depicted in Figure 5b, uses an analogous idea to force Γ to approach the center of the gadget no matter if it passed through ${}^{\rightarrow}◣_{i,t}$ for $t = 0$ or $t = 1$. We combine different variable gadgets via entrance and exit areas, indicated for the ith variable as ${}^{\rightarrow}◗_i$ and $◗_i^{\rightarrow}$, and highlight two points s_i and t_i inside them that will act as a certificate necessary in the full proof. We define the start area for ${}^{\rightarrow}◗_1$ as s and, similarly, the end area for $◗_N^{\rightarrow}$ as t. Note that in Figure 5 the gadget for x_1 is closed around s_1; we do so likewise for x_N. Finally, the second component consists of multiple tunnels that connect the first component via the gadgets for clauses c_j with $x_i \in c_j$ to the third component of the gadget. For negative clauses, we attach to them half of a tunnel to toggle the configuration and thus its carried truth state before visiting these clauses.

Complete Reduction. The placement of the obstacles in our gadgets ensures that any configuration follows pre-determined paths through the gadgets. Formally, we have:

▶ **Lemma 1.** *Let $\mathcal{L}$ be a unit-length linear linkage with a configuration Γ of $\mathcal{L}$ that passes through a gadget F with an entrance and exit pair for the variable $x_i \in \mathcal{X}$. If there exists*

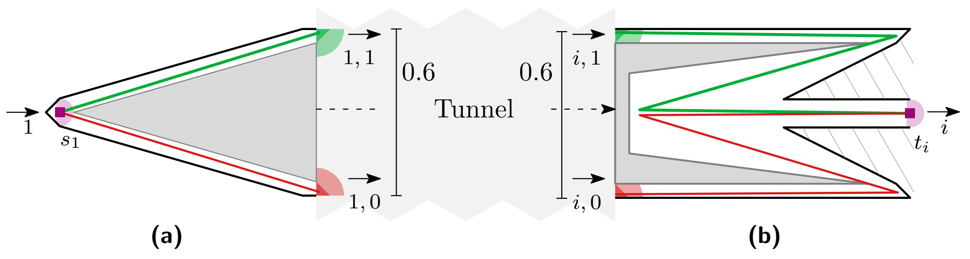


Figure 5 The **(a)** first and **(b)** third component of the variable gadget for x_1 and x_i, respectively.

a vertex $v \in V$ such that $\Gamma(v) \in {}^{\rightarrow}\!◣_{i,t}(F)$ for $t \in \{0,1\}$, we have $\Gamma(v') \in ◣^{\rightarrow}_{i,t}(F)$ for some $v' \in V$. Furthermore, any configuration Γ' of $\mathcal{L}$ can be perturbed such that every vertex inside some entrance/exit area will lie in the corresponding start/end area.

We now use Lemma 1 to combine the above-introduced gadgets by taking the planar rectilinear drawing $\mathcal{E}$ of the incidence graph of φ and replacing all components with the respective gadgets. Observe that we can scale (parts of) $\mathcal{E}$ by appropriate polynomial factors to ensure that there is enough space for the placement of the gadgets. Hence, we can ensure that the entrance and exit areas of the respective gadgets coincide and can, furthermore, unify all vertices in different gadgets that lie on the same point. To finish the construction of $(\mathcal{L}, P, s, t)$, we close the first and last variable gadget and set $s = s_1$ and $t = t_N$. We note that the obtained polygonal domain P contains (polynomially many) obstacles, i.e., holes. Let P be created using T tunnels, excluding those used in other gadgets such as the shifters, and B bends. The linear linkage $\mathcal{L}$ consists of $n = 2T + 7B + 4\,|\mathcal{X}| + 35\,|\mathcal{C}| + 2\,|\mathcal{C}^-| + 1$ vertices. When carefully analyzing (the construction of) our gadgets, we observe that any planar configuration Γ of $\mathcal{L}$ starting at s must pass through every gadget exactly once. Otherwise it is too short to reach t. Using Observation 1, we conclude that for Γ to be planar, every clause must be satisfied, establishing NP-hardness of ULLLR.

4 Concluding Remarks and Future Directions

We see this abstract as a further step towards understanding the complexity of realizing linkages in polygonal domains. Our hardness result from Theorem 1 raises the question in which settings we can solve ULLLR in polynomial time. For very restricted instances, an efficient algorithm is indeed possible, as our last result shows:

▶ **Theorem 2.** ULL LINKAGE REALIZABILITY *for three-edge linkages in a convex polygon P with m vertices can be solved in time $\mathcal{O}(m)$ even for general linear linkages.*

Theorem 2 hinges on three-edge linkages having only one degree of freedom and an extension to four-edge linkages is highly non-trivial. We see a generalization to arbitrary constant-size instances of ULLLR as a direction for future work. Furthermore, $\exists\mathbb{R}$-hardness for general, or NP-hardness for simple/convex polygonal domains are interesting directions to pursue.

References

1 Zachary Abel, Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Jayson Lynch, and Tao B. Schardl. Who Needs Crossings? Hardness of Plane Graph Rigidity. In Sándor P. Fekete and Anna Lubiw, editors, *Proc. 32nd International Symposium on Computational Geometry (SoCG'16)*, volume 51 of *LIPIcs*, pages 3:1–3:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. `doi:10.4230/LIPICS.SOCG.2016.3`.

2 Helmut Alt, Christian Knauer, Günter Rote, and Sue Whitesides. The complexity of (un)folding. In Steven Fortune, editor, *Proc. 19th International Symposium on Computational Geometry (SoCG'03)*, pages 164–170. ACM, 2003. `doi:10.1145/777792.777818`.

3 Therese Biedl, Erik Demaine, Martin Demaine, Sylvain Lazard, Anna Lubiw, Joseph O'Rourke, Steve Robbins, Ileana Streinu, Godfried Toussaint, and Sue Whitesides. A note on reconfiguring tree linkages: trees can lock. *Discrete Applied Mathematics*, 117(1–3):293–297, 2002. `doi:10.1016/s0166-218x(01)00229-3`.

4 Sergio Cabello, Erik D. Demaine, and Günter Rote. Planar Embeddings of Graphs with Specified Edge Lengths. In Giuseppe Liotta, editor, *Proc. 11th International Symposium on Graph Drawing and Network Visualization (GD'03)*, volume 2912 of *LNCS*, pages 283–294. Springer, 2004. `doi:10.1007/978-3-540-24595-7_26`.

5 Sergio Cabello, Erik D. Demaine, and Günter Rote. Planar Embeddings of Graphs with Specified Edge Lengths. *Journal of Graph Algorithms and Applications*, 11(1):259–276, 2007. `doi:10.7155/jgaa.00145`.

6 Robert Connelly, Erik D. Demaine, and Günter Rote. Straightening Polygonal Arcs and Convexifying Polygonal Cycles. *Discrete & Computational Geometry*, 30(2):205–239, 2003. `doi:10.1007/S00454-003-0006-7`.

7 Mark de Berg and Amirali Khosravi. Optimal Binary Space Partitions for Segments in the Plane. *International Journal of Computational Geometry and Application*, 22(03):187–205, 2012. `doi:10.1142/s0218195912500045`.

8 Hubert de Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990. `doi:10.1007/bf02122694`.

9 Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, 2007. `doi:10.1017/cbo9780511735172`.

10 István Fáry. On straight-line representation of planar graphs. *Acta Sci. Math. (Szeged)*, 11:229—233, 1948.

11 John Hopcroft, Deborah Joseph, and Sue Whitesides. Movement Problems for 2-Dimensional Linkages. *SIAM Journal on Computing*, 13(3):610–629, 1984. `doi:10.1137/0213038`.

12 John Hopcroft, Deborah Joseph, and Sue Whitesides. On the Movement of Robot Arms in 2-Dimensional Bounded Regions. *SIAM Journal on Computing*, 14(2):315–333, 1985. `doi:10.1137/0214025`.

13 Chenhua Huang, Xiangbo Yang, and Zhihong He. Protein folding simulations of 2D HP model by the genetic algorithm based on optimal secondary structures. *Computational Biology and Chemistry*, 34(3):137–142, 2010. `doi:10.1016/j.compbiolchem.2010.04.002`.

14 Deborah A. Joseph and W. Harry Plantings. On the Complexity of Reachability and Motion Planning Questions (Extended Abstract). In *Proc. 1st International Symposium on Computational Geometry (SoCG)*, pages 62–66. ACM, 1985. `doi:10.1145/323233.323242`.

15 Vitit Kantabutra. Motions of a short-linked robot arm in a square. *Discrete & Computational Geometry*, 7(1):69–76, 1992. `doi:10.1007/bf02187825`.

16 Michael Kapovich and John J. Millson. Universality theorems for configuration spaces of planar linkages. *Topology*, 41(6):1051–1107, 2002. `doi:10.1016/s0040-9383(01)00034-9`.

17 Alfred B. Kempe. On a General Method of describing Plane Curves of the nth degree by Linkwork. *Proceedings of the London Mathematical Society*, 1(1):213–216, 1875. `doi:10.1112/plms/s1-7.1.213`.

18 Joseph S. B. Mitchell. Geometric Shortest Paths and Network Optimization. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, chapter 15, pages 633–701. Elsevier, 2000. `doi:10.1016/b978-044482537-7/50016-4`.

19 Walter Schnyder. Embedding Planar Graphs on the Grid. In David S. Johnson, editor, *Proc. 1st ACM-SIAM Symposium on Discrete Algorithms (SODA'90)*, pages 138–148. SIAM, 1990.

20 Marc J. van Kreveld, Jack Snoeyink, and Sue Whitesides. Folding Rulers Inside Triangles. *Discrete & Computational Geometry*, 15(3):265–285, 1996. `doi:10.1007/bf02711495`.

21 Klaus Wagner. Bemerkungen zum Vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936. URL: `http://eudml.org/doc/146109`.

22 Sue Whitesides and Naixun Pei. On the Reconfiguration of Chains (Extended Abstract). In Jin-yi Cai and C. K. Wong, editors, *Proc. 2nd International Computing and Combinatorics Conference (COCOON'96)*, volume 1090 of *Lecture Notes in Computer Science*, pages 381–390. Springer, 1996. `doi:10.1007/3-540-61332-3_172`.

Crossing Number of 3-Plane Drawings*

Miriam Goetze†1, Michael Hoffmann2, Ignaz Rutter‡3, and Torsten Ueckerdt1

1 **Karlsruhe Institute of Technology, Germany**
`miriam.goetze@kit.edu`, `torsten.ueckerdt@kit.edu`

2 **Department of Computer Science, ETH Zürich, Switzerland**
`hoffmann@inf.ethz.ch`

3 **University of Passau, Germany**
`rutter@fim.uni-passau.de`

Abstract

We study 3-plane drawings, that is, drawings of graphs in which every edge has at most three crossings. We show how the recently developed Density Formula for topological drawings of graphs [9] can be used to count the crossings in terms of the number n of vertices. As a main result, we show that every 3-plane drawing has at most $5.5(n-2)$ crossings, which is tight. In particular, it follows that every 3-planar graph on n vertices has crossing number at most $5.5n$, which improves upon a recent bound [3] of $6.6n$. To apply the Density Formula, we carefully analyze the interplay between certain configurations of cells in a 3-plane drawing. As a by-product, we also obtain an alternative proof for the known statement that every 3-planar graph has at most $5.5(n-2)$ edges.

1 Introduction

One of the most basic combinatorial questions one can ask for a class of graphs is: How many edges can a graph from this class have as a function of the number n of vertices? Prominent examples include upper bounds of $\binom{n}{2}$ for the class of all graphs and $\frac{n^2}{4}$ for bipartite graphs. These bounds are immediate consequences of the definition of these graph classes, and they are tight, that is, there exist graphs in the class with exactly this many edges. But for several other graph classes good upper bounds on the number of edges are much more challenging to obtain. Notably this holds for classes that relate to the existence of certain geometric representations. One the most fundamental questions one can ask about a class of geometrically represented graphs is: What is the minimum number of edge crossings required in such a representation, as a function of the number n of vertices? We study both of these fundamental questions in combination, for the class of 3-planar graphs. A graph is *k-planar* if it can be drawn in the plane such that every edge has at most k crossings. The study of k-planar graphs goes back to Ringel [16] and has been a major focus in graph drawing over the past two decades [8], as a natural generalization of planar graphs ($k = 0$).

The maximum number of edges in a simple k-planar graph on n vertices is known to be at most $c_k(n-2)$, where $c_0 = 3$, $c_1 = 4$ [5], $c_2 = 5$ [14, 15], $c_3 = 5.5$ [10, 11], $c_4 = 6$ [1], and $c_k \leq 3.81\sqrt{k}$, for general $k \geq 5$ [1]. The bounds for $k \leq 2$ are tight and those for $k \leq 4$ are tight up to an additive constant [1, 4]. The bounds for $k \leq 4$ also generalize to *non-homotopic* drawings of multigraphs [12, 13], that is, where every continuous transformation

* This research was initiated at the Workshop on *Graph and Network Visualization* (GNV 2024) in Heiligkreuztal, Germany, June 23–28, 2024.
† funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 520723789
‡ funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 541433306

that transforms one copy of an edge to another passes over a vertex. Interestingly, the upper bound for 3-planar graphs is tight in this more general setting only [4, 6].

The *crossing number* of a drawing Γ is the number of edge crossings in Γ. The *crossing number* $\mathrm{cr}(G)$ of a graph G is the minimum crossing number over all drawings of G. By definition every k-planar graph G admits a k-plane drawing and thus

$$\mathrm{cr}(G) \leq \frac{km}{2}, \tag{S}$$

where m denotes the number of edges in G. For a k-planar graph, this simple inequality connects upper bounds on the number of edges with lower bounds on the crossing number. Both of these come together in the well-known Crossing Lemma [2, Chapter 45], as the best constants in the Crossing Lemma are obtained by analyzing k-plane drawings [1, 6, 10, 11]. Conversely, combining the lower bound on $\mathrm{cr}(G)$ from the Crossing Lemma with an upper bound on $\mathrm{cr}(G)$ we obtain an upper bound on the number of edges in G. While (S) would work here, it is probably not an ideal choice because the graphs for which (S) is tight might be very different from those graphs that have a maximum number of edges, for any fixed n. For instance, for a 1-planar graph G we have $\mathrm{cr}(G) \leq n - 2$ [17, Proposition 4.4], which beats the bound we get by plugging $m \leq 4n - 8$ into (S) by a factor of two. Can we obtain similar improvements by bounding $\mathrm{cr}(G)$ in terms of n, rather than m, for $k \geq 2$?

Indeed, very recently it has been shown that $\mathrm{cr}(G) \leq 3.\overline{3}n$ if G is 2-planar and $\mathrm{cr}(G) \leq 6.6n$ if G is 3-planar [3]. There is some indication that the bound for 2-planar graphs could be tight up to an additive constant, as it is achieved by the standard drawings of optimal 2-planar graphs (Figure 1). But the crossing number of these graphs is not known.

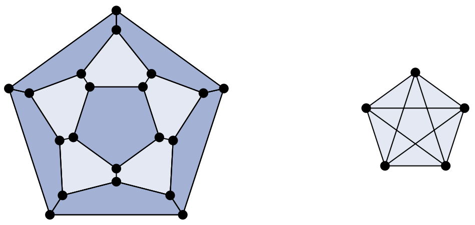


Figure 1 Construction by Pach and Tóth [15, Figure 3]. Left: A planar drawing with pentagonal faces. Right: To each pentagonal face all diagonals are added.

In contrast, there exists a family of simple 3-planar graphs with $5.5n - 15$ edges whose standard drawings have $5.5n - 21$ crossings (Figure 2). Thus, there is a gap of $1.1n$ between the lower and the upper bound for the crossing number of 3-plane drawings.

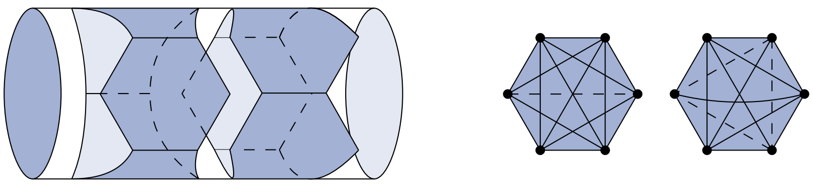


Figure 2 Construction from [11, Figure 8]. Left: A cylinder with two layers, each consisting of three hexagonal faces. Right: To each face of a layer all but one diagonal is added. To the top and bottom face six diagonals are added. Missing diagonals are represented by dashed lines.

Results. We close the gap and present an upper bound on the crossing number of 3-plane drawings that is tight up to an additive constant. Using the same approach we also obtain an alternative proof to show that a 3-planar n-vertex graph has at most $5.5(n-2)$ edges.

▶ **Theorem 1.** *Every non-homotopic 3-plane drawing of a graph on n vertices, $n \geq 3$, contains at most $5.5(n-2)$ edges and at most $5.5(n-2)$ crossings.*

Our proof relies on the recently developed Density Formula (cf. Theorem 2 below) for topological drawings of graphs [9]. It relates the number of vertices, edges, and cells of various sizes in a drawing, in a way similar to the Euler Formula in the case of plane graphs. Previously, the Density Formula has been used to derive upper bounds on the number of edges in k-plane drawings, for $k \leq 2$ [9]. In order to apply it to 3-plane drawings, to bound the number of crossings, and to obtain tight bounds, we study cells not only in isolation but also as part of what we call *configurations*, which consist of several connected cells. We then develop a number of new constraints that relate the number of cells and/or configurations of a certain type in any 3-plane drawing. The combination of all these constraints with the Density Formula yields a linear program that we can solve in two different ways—maximizing either the number of edges or the number of crossings—to prove Theorem 1.

Using Theorem 1 we can derive better upper bounds on the number of edges in k-planar graphs without short cycles. Plugging our bound of at most $5.5n$ crossings into the proofs from [3] we obtain that

- C_3-free 3-planar graphs on n vertices have at most $\sqrt[3]{891/8}n < 4.812n$ edges (down from $\approx 5.113n$ [3, Theorem 18]),
- C_4-free 3-planar graphs on n vertices have at most $\sqrt[3]{1'254'825/12'544}n < 4.643n$ edges (down from $\approx 4.933n$ [3, Theorem 20]), and
- 3-planar graphs of girth 5 on n vertices have at most $\sqrt[3]{122,793/1600}n < 4.25n$ edges (down from $\approx 4.516n$ [3, Theorem 21]).

2 Preliminaries

We consider *drawings* of graphs on the sphere with vertices as points, edges as Jordan arcs, and the usual assumption that any two edges share only finitely many points, each being a common endpoint or a proper crossing, and that no three edges cross in the same point. We also assume that no edge crosses itself and that no two adjacent edges cross. As is customary, we do not distinguish between the points and curves in Γ and the vertices and edges of G they represent, respectively. The graphs we consider may contain parallel edges, but no loops. In order to avoid an arbitrary number of parallel edges within a small corridor, a drawing Γ is called *non-homotopic* if every region that is bounded by exactly two parts of edges, called a *lens*, contains a crossing or a vertex in its interior; see Figure 3.

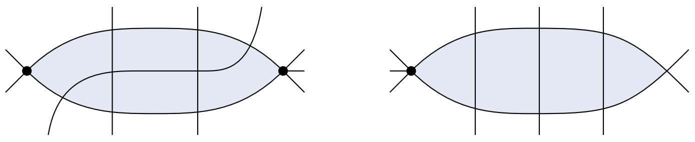


Figure 3 *Left:* A lens (blue) with two crossings in its interior. *Right:* An empty lens (blue).

Let Γ be a drawing of a graph $G = (V, E)$. If every edge is crossed at most three times, we say that Γ is 3-*plane*. We denote the set of crossings by X. For $i \in \{0, 1, 2, 3\}$, let $E_i \subseteq E$ be the set of all edges with exactly i crossings, and let $E_\times = E_1 \cup E_2 \cup E_3$.

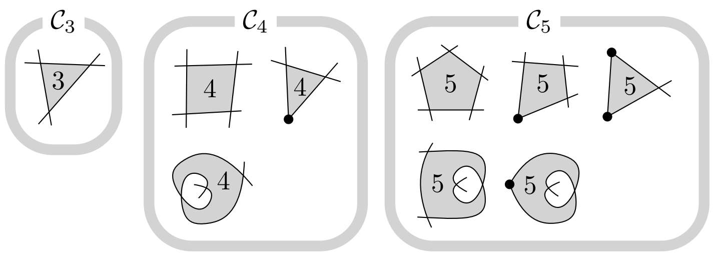


Figure 4 Taken from [9, Figure 2]. All types of cells c of size $\|c\| \leq 5$ in a non-homotopic connected drawing on at least three vertices. The bottom row shows the degenerate cells.

Edge-Segments and Cells. An edge with i crossings is split into $i+1$ parts, called *edge-segments*. An edge-segment is *inner* if both its endpoints are crossings, and *outer* otherwise. The *planarization* of Γ is the graph obtained by replacing every crossing x with a vertex of degree 4 that is incident to the four edge-segments of x. We say that the drawing Γ is *connected*, if its planarization is a connected graph, and shall henceforth only consider connected drawings. Removing all edges and vertices of Γ splits the sphere into several components, called *cells*. We denote the set of all cells by $\mathcal{C}$. Since Γ is connected, the *boundary* ∂c of a cell c corresponds to a cyclic sequence alternating between edge-segments and elements in $V \cup X$ (i.e., vertices and crossings). If a crossing or a vertex appears multiple times on the boundary of the same cell c, then c is *degenerate*. The *size* of a cell c, denoted by $\|c\|$, is the number of vertex incidences plus the number of edge-segment incidences of c. Note that incidences with crossings are not taken into account, see Figure 4 for examples. For $a \in \mathbb{N}$, we denote by $\mathcal{C}_a = \{c \in \mathcal{C} \colon \|c\| = a\}$ the set of all cells of size a.

▶ **Theorem 2** (Density Formula [9]). *If Γ is a connected drawing with at least one edge, and t is a real number, then*

$$|E| = t(|V| - 2) - \sum_{c \in \mathcal{C}} \left(\frac{t-1}{4} \|c\| - t \right) - |X|$$

To apply the Density Formula, we count the cells of different sizes. We distinguish several types of cells based on their size and boundary and denote these by small pictograms, such as △ or $\boxed{5}$. We call a cell *large* if it has size at least 6 and write ○ for this type of cells. By abuse of notation, we denote the number of cells of a certain type by their pictogram.

Configurations are connected labeled embedded subgraphs of the planarization of a drawing Γ. We denote configuration types by pictograms such as △ and △ (see Figure 5).

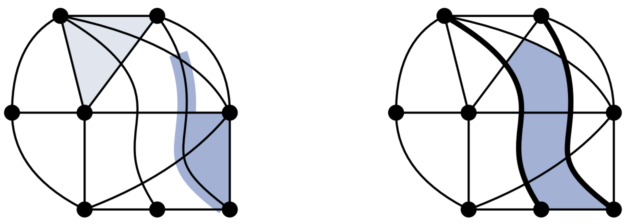


Figure 5 Left: A △-configuration (light blue) and a △-configuration (dark blue). Right: A ⬠-○-trail (dark blue) and its bounding edges (thick).

A configuration is an *A-B-trail* if its dual is a path P whose endpoints are cells of type $A \neq \boxed{4}$ and $B \neq \boxed{4}$, respectively, whose edges correspond to inner segments, and whose interior vertices are $\boxed{4}$-cells whose two edge-segments on P are opposite along their boundary, see Figure 5. We denote by $(A \leftrightarrow B)$ the number of A-B-trails in Γ.

▶ **Observation 3.** *Every inner edge-segment of a drawing is interior to exactly one trail.*

A drawing is *filled* if any two vertices $u \neq v$ on the boundary of a cell c are joined by an uncrossed edge along ∂c. A 3-plane, non-homotopic, connected, filled drawing of a graph on at least three vertices is 3-*saturated.*

3 Crossing-Number and Edge-Density via Density Formula

To obtain our upper bounds we prove a number of (in)equalities, each relating the number of certain cells, configurations, edges and crossings. The Density Formula is one such equality. In total, we obtain a system of linear inequalities where each quantity (such as $|E|$, $(|V|-2)$, $|X|$, $|\mathcal{C}_2|$, $|E_1|$, $\triangledown_3$, $\bigcirc$, etc.) can be considered as a variable. Setting the "variable" $(|V|-2)$ to 1, we can maximize the value of $|X|$ by solving the obtained linear program (LP). The resulting maximum represents the number of crossings per vertex; more precisely, per $(|V|-2)$. We want to prove that the number of crossings in any 3-plane drawing on n vertices is at most $5.5(n-2)$. It thus suffices to show that the maximum value of $|X|$ in the LP is 5.5 if we set the variable representing the number of vertices to 1. Our LP comprises 21 constraints, which are summarized in Figure 6. The validity of two constraints (namely (3.C) and (5.A)) is proven in Section 4. Constraints that are only proven in the full version are marked with $(\star)$. Summing up all constraints with the coefficients in Figure 6, we obtain $|X| \leq 5.5(|V|-2)$.

If we maximize $|E|$ instead, we obtain $|E| \leq 5.5(|V|-2)$ from the same constraints (with different coefficients; also in Figure 6). Hence, by verifying that all 21 constraints hold for every connected, non-homotopic 3-plane drawing on $n \geq 3$ vertices, we obtain our result.

▶ **Theorem 1.** *Every non-homotopic* 3*-plane drawing of a graph on* n *vertices,* $n \geq 3$*, contains at most* $5.5(n-2)$ *edges and at most* $5.5(n-2)$ *crossings.*

4 Relating Crossing, Edge, Cell, Trail, and Configuration Counts

In this section, we present a number of (in)equalities, each relating the number of certain cells, configurations, edges, or crossings. Due to space constraints we discuss only two of these inequalities, the rest can be found in the full version. Our proof relies on the Density Formula for $t = 5$. For this value of t, $\bigcirc$-cells contribute negatively in the formula. Intuitively, large cells account for many crossings: If many trails end in large cells, we obtain a lower bound on the sum $\sum_{a\geq 6} a|\mathcal{C}_a|$ of sizes of large cells. This yields a lower bound on the sum $\sum_{c\in\mathcal{C}_{\geq 6}}(\|c\|-5)$ in the Density Formula, where $\mathcal{C}_{\geq 6}$ denotes the set of large cells. If there are few such trails, we obtain configurations that contain many crossed edges.

▶ **Lemma 4.** *If* Γ *is a* 3*-saturated drawing, then*

$$\sum_{a\geq 6} a|\mathcal{C}_a| \geq (\triangle_4 \leftrightarrow \bigcirc) + (\boxed{5} \leftrightarrow \bigcirc) + (\triangledown_3 \leftrightarrow \bigcirc) + (⬠_5 \leftrightarrow \bigcirc) + 5\,\boxed{6}. \qquad (5.A)$$

Proof. As we want to obtain a lower bound on the sum $\sum_{a\geq 6} a|\mathcal{C}_a|$, it suffices to count the number of vertex and edge-segment incidences of large cells. Each trail that ends in a large cell enters this cell via an inner edge-segment. As no two trails share such an inner edge-segment, we obtain one edge-segment incidence for each such trail.

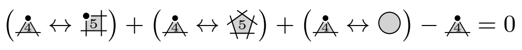
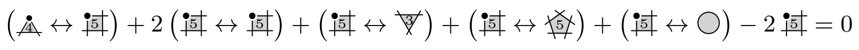
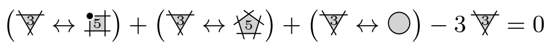
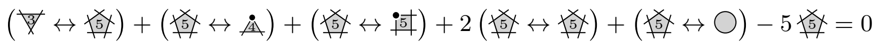
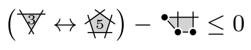
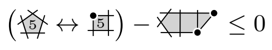
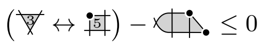
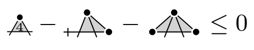
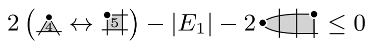
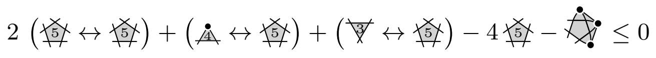
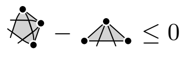
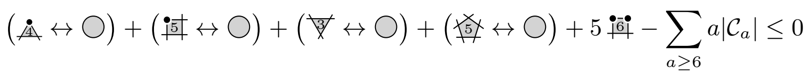
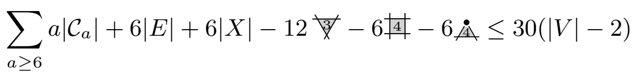
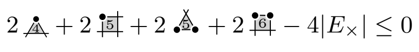
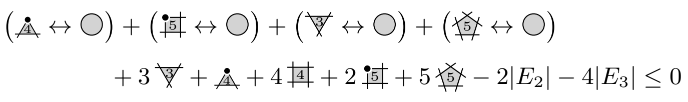
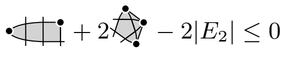
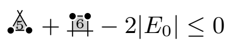
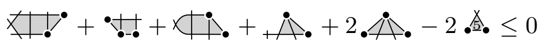
	(In)equality	$\lvert E\rvert$	$\lvert X\rvert$
(⋆)		$\frac{-5}{16}$	$\frac{-7}{16}$
(⋆)		$\frac{5}{16}$	$\frac{5}{16}$
(⋆)		$\frac{-11}{24}$	$\frac{-11}{24}$
(⋆)		$\frac{1}{8}$	$\frac{-3}{8}$
(⋆)		$\frac{7}{48}$	$\frac{1}{48}$
(⋆)		0	$\frac{1}{16}$
(3.C)		$\frac{3}{16}$	$\frac{7}{48}$
(⋆)		$\frac{3}{16}$	$\frac{5}{16}$
(⋆)		0	$\frac{1}{16}$
(⋆)		$\frac{3}{16}$	$\frac{13}{16}$
(⋆)		$\frac{3}{16}$	$\frac{5}{16}$
(5.A)		$\frac{11}{60}$	$\frac{11}{60}$
(⋆)		$\frac{11}{60}$	$\frac{11}{60}$
(⋆)		$\frac{13}{80}$	$\frac{3}{80}$
(⋆)		$\frac{11}{40}$	$\frac{11}{40}$
(⋆)	$\lvert E_1\rvert + \lvert E_2\rvert + \lvert E_3\rvert - \lvert E_\times\rvert = 0$	$\frac{-11}{20}$	$\frac{19}{20}$
(⋆)	$\lvert E_1\rvert + 2\lvert E_2\rvert + 3\lvert E_3\rvert - 2\lvert X\rvert = 0$	$\frac{11}{20}$	$\frac{1}{20}$
(⋆)		0	$\frac{1}{4}$
(⋆)	$\lvert E_\times\rvert + \lvert E_0\rvert - \lvert E\rvert = 0$	$\frac{1}{10}$	$\frac{11}{10}$
(⋆)		$\frac{1}{20}$	$\frac{11}{20}$
(⋆)		$\frac{3}{16}$	$\frac{5}{16}$

Figure 6 Certificates for the upper bound on the number of edges and crossings in 3-saturated drawings in terms of the number of vertices. Each row corresponds to one inequality. In order to obtain the upper bound on the number of edges, we multiply each inequality with the third entry in the corresponding row and sum up all the inequalities. To obtain the upper bound on the number of crossings we proceed likewise using the fourth entry of each row as a coefficient.

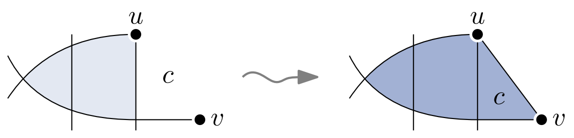


Figure 7 A $\boxed{3}$-$\boxed{5}$-trail (light blue). It forms a $\boxed{\cdot}$-configuration (dark) with an adjacent cell.

A $\boxed{6}$-cell is in particular large. As it is incident to only one inner-segment, it is the endpoint of only one trail. We have not counted the remaining three edge-segment incidences and the two vertex incidences when considering trails. Therefore, each $\boxed{6}$-cell yields at least five more edge-segment and vertex incidences. ◀

▶ **Lemma 5.** *If Γ is a 3-saturated drawing, then*

$$(\boxed{3} \leftrightarrow \boxed{5}) \leq \boxed{\cdot}. \tag{3.C}$$

Proof. Consider a $\boxed{3}$-$\boxed{5}$-trail. As every edge is crossed at most three times, the trail contains no $\boxed{4}$-cell and we are in the situation represented in Figure 7. The vertices u and v lie on the boundary of a cell c. As the drawing is 3-saturated, the edge uv is contained in G and the cell c is a $\boxed{2}$-cell. The trail together with c forms a $\boxed{\cdot}$-configuration. As every $\boxed{3}$-$\boxed{5}$-trail is only part of one such configuration, the statement follows. ◀

5 Discussion

The *k-planar crossing number* $\mathrm{cr}_k(G)$ is similar to the crossing number, except that the minimum is taken over all k-plane drawings of G. Clearly, $\mathrm{cr}(G) \leq \mathrm{cr}_k(G)$ for all k and G. But there are k-planar n-vertex graphs G with $\mathrm{cr}(G) \in \mathcal{O}(k)$ and $\mathrm{cr}_k(G) \in \Omega(kn)$ [7, Theorem 2]. By Theorem 1, every 3-plane drawing of an n-vertex graph G has $|X| \leq 5.5(n-2)$ crossings, and hence $\mathrm{cr}(G) \leq \mathrm{cr}_3(G) \leq 5.5(n-2)$. Although Theorem 1 is tight, we could have $\mathrm{cr}(G), \mathrm{cr}_3(G) < 5.5(n-2)$, and a similar question arises for 2-planar graphs.

▶ **Question 6.**

- *Are there 3-planar n-vertex graphs G with $\mathrm{cr}_3(G) = 5.5(n-2)$ or $\mathrm{cr}(G) = 5.5(n-2)$?*
- *Are there 2-planar n-vertex graphs G with $\mathrm{cr}_2(G) = 3.\overline{3}(n-2)$ or $\mathrm{cr}(G) = 3.\overline{3}(n-2)$?*

References

1 Eyal Ackerman. On topological graphs with at most four crossings per edge. *Computational Geometry*, 85:101574, 2019. doi:10.1016/j.comgeo.2019.101574.

2 Martin Aigner and Günter M. Ziegler. *Proofs from THE BOOK*. Springer, Berlin, 6th edition, 2018. doi:10.1007/978-3-662-57265-8_45.

3 Michael A. Bekos, Prosenjit Bose, Aaron Büngener, Vida Dujmović, Michael Hoffmann, Michael Kaufmann, Pat Morin, Saeed Odak, and Alexandra Weinberger. On k-planar graphs without short cycles. In *Proc. 32nd Internat. Sympos. Graph Drawing Network Visualization (GD 2024)*, volume 320 of *LIPIcs*, pages 27:1–27:17, 2024. doi:10.4230/LIPIcs.GD.2024.27.

4 Michael A. Bekos, Michael Kaufmann, and Chrysanthi N. Raftopoulou. On optimal 2-and 3-planar graphs. In *33rd International Symposium on Computational Geometry (SoCG 2017)*, 2017. doi:10.4230/LIPIcs.SoCG.2017.16.

5 Rainer Bodendiek, Heinz Schumacher, and Klaus Wagner. Bemerkungen zu einem Sechsfarbenproblem von G. Ringel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 53:41–52, 1983. doi:10.1007/BF02941309.

6 Aaron Büngener and Michael Kaufmann. Improving the Crossing Lemma by characterizing dense 2-planar and 3-planar graphs. In *Proc. 32nd Internat. Sympos. Graph Drawing Network Visualization (GD 2024)*, volume 320 of *LIPIcs*, pages 29:1–29:22, 2024. URL: `https://doi.org/10.4230/LIPIcs.GD.2024.29`, `doi:10.4230/LIPICS.GD.2024.29`.

7 Markus Chimani, Torben Donzelmann, Nick Kloster, Melissa Koch, Jan-Jakob Völlering, and Mirko H. Wagner. Crossing numbers of beyond planar graphs re-revisited: A framework approach. In *32nd Internat. Sympos. Graph Drawing Network Visualization (GD 2024)*, volume 320 of *LIPIcs*, pages 33:1–33:17, 2024. `doi:10.4230/LIPIcs.GD.2024.33`.

8 Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. A survey on graph drawing beyond planarity. *ACM Comput. Surv.*, 52(1):1–37, 2020. `doi:10.1145/3301281`.

9 Michael Kaufmann, Boris Klemz, Kristin Knorr, Meghana M. Reddy, Felix Schröder, and Torsten Ueckerdt. The density formula: One lemma to bound them all. In *32nd Internat. Sympos. Graph Drawing Network Visualization(GD 2024)*, volume 320 of *LIPIcs*, pages 7:1–7:17, 2024. `doi:10.4230/LIPIcs.GD.2024.7`.

10 János Pach, Radoš Radoičić, Gábor Tardos, and Géza Tóth. Improving the Crossing Lemma by finding more crossings in sparse graphs. In *Proc. 20th Annu. Sympos. Comput. Geom. (SoCG 2004)*, pages 68–75, 2004. `doi:10.1145/997817.997831`.

11 János Pach, Radoš Radoičić, Gábor Tardos, and Géza Tóth. Improving the Crossing Lemma by finding more crossings in sparse graphs. *Discrete Comput. Geom.*, 36(4):527–552, 2006. `doi:10.1007/s00454-006-1264-9`.

12 János Pach, Gábor Tardos, and Géza Tóth. Crossings between non-homotopic edges. In *Proc. 28th Internat. Sympos. Graph Drawing Network Visualization (GD 2020)*, volume 12590 of *LNCS*, pages 359–371, 2020. `doi:10.1007/978-3-030-68766-3\_28`.

13 János Pach, Gábor Tardos, and Géza Tóth. Crossings between non-homotopic edges. *J. Comb. Theory B*, 156:389–404, 2022. URL: `https://doi.org/10.1016/j.jctb.2022.05.007`, `doi:10.1016/J.JCTB.2022.05.007`.

14 János Pach and Géza Tóth. Graphs drawn with few crossings per edge. In *Proc. 4th Internat. Sympos. Graph Drawing (GD 1996)*, volume 1190 of *LNCS*, pages 345–354, 1996. `doi:10.1007/3-540-62495-3\_59`.

15 János Pach and Géza Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17(3):427–439, 1997. `doi:10.1007/BF01215922`.

16 Gerhard Ringel. Ein Sechsfarbenproblem auf der Kugel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 29:107–117, 1965. `doi:10.1007/BF02996313`.

17 Yusuke Suzuki. 1-planar graphs. In *Beyond Planar Graphs, Communications of NII Shonan Meetings*, pages 47–68. Springer, 2020. `doi:10.1007/978-981-15-6533-5\_4`.

Polycube Segmentations via Dual Loops

Maxim Snoep, Bettina Speckmann, and Kevin Verbeek

TU Eindhoven, The Netherlands
`[m.snoep|b.speckmann|k.a.b.verbeek]@tue.nl`

Abstract

Polycube segmentations for 3D models effectively support a wide variety of applications such as hexahedral mesh construction, seamless texture mapping, spline fitting, and multi-block grid generation. However, the automated construction of valid polycube segmentations suffers from robustness issues: state-of-the-art methods are not guaranteed to find a valid solution. In this paper we present an iterative algorithm which is guaranteed to return a valid polycube segmentation for 3D models of any genus. Our algorithm is based on a novel dual representation of polycubes [Snoep, Speckmann, & Verbeek, 2025]. Starting from an initial simple polycube of the correct genus, together with the corresponding dual loop structure and polycube segmentation, we iteratively refine the polycube, loop structure, and segmentation, while maintaining the correctness of the solution. Our algorithm is robust by construction: at any point during the iterative process the current segmentation is valid. Furthermore, the iterative nature of our algorithm facilitates a seamless trade-off between quality and complexity of the solution.

1 Introduction

Polycubes are orthogonal polyhedra with axis-aligned quadrilateral faces. The simple structure of polycubes enables efficient solutions to various challenging geometric problems. Bijective mappings from general shapes to polycubes, known as *polycube maps*, enable the transfer of solutions computed on polycubes to those general shapes. Polycube maps are used to solve problems such as texture mapping [7], spline fitting [8], and hexahedral meshing [4]. Formally, a polycube map f is a continuous map from a polycube Q of genus g to a closed 2-dimensional surface $\mathcal{M}$ of the same genus. The edges of Q map to a segmentation of $\mathcal{M}$ into *patches* that correspond to the faces of Q, known as a *polycube segmentation* (see Figure 1).

Current methods for constructing polycube segmentations [2,3] are not robust: they are not guaranteed to return a valid solution. In this paper we describe an iterative robust algorithm

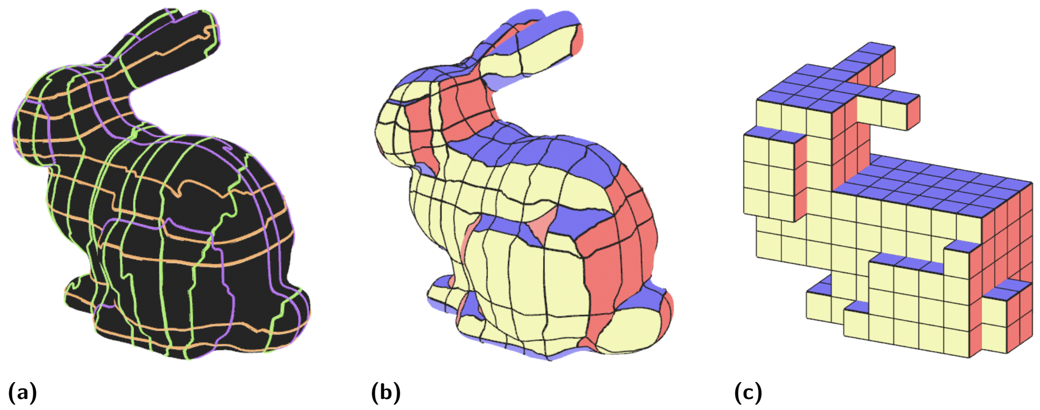


Figure 1 A polycube loop structure (a) with its polycube segmentation (b) and polycube (c).

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

to construct polycube segmentations. Our algorithm is based on a recent characterization of polycubes via a dual loop structure [5]. In Section 2 we review the necessary background and then describe our algorithm in Section 3. Section 4 showcases some results from a proof-of-concept implementation.

2 Preliminaries

Our algorithm is based on the characterization of polycubes by Snoep, Speckmann, and Verbeek [5]. In this section we review the necessary definitions and results from their work.

A *quadrilateral mesh (quad mesh)* consists of vertices, edges, and quadrilateral faces. Each vertex is adjacent to at least one edge. Each edge is adjacent to one or two faces. Each face consists of four vertices and four edges. A quad mesh is *closed* if each edge is adjacent to exactly two faces. A quad mesh is *orientable* if a consistent circular ordering of vertices can be assigned to each face, such that edge-adjacent faces have opposite vertex orders along their common edge. A quad mesh is *connected* if every vertex can be reached from any other vertex by traversing edges. We can now define a polycube (see Figure 2).

▶ **Definition 2.1.** A *polycube* Q is a closed, connected, orientable quad mesh with vertices $V(Q)$ such that:

1. Each vertex $v \in V(Q)$ has a position $p(v)$ in $\mathbb{Z}^3$,
2. Each vertex has degree at least 3,
3. Positions of adjacent vertices differ in exactly one coordinate,
4. Edges incident to the same vertex cannot overlap.

As a consequence, our polycubes have vertices with degrees up to six (see Figure 2a). The polycube faces are not required to be unit squares (see Figure 2b), as this may not be general enough for higher genus polycubes (see Figure 2c). According to this definition, polycubes are also allowed to self-intersect, since this does not pose a problem for surface maps [6] (see Figure 2d). Note that self-intersections might cause problems for volumetric methods, such as hex meshing.

Each polycube defines three partial orders on its vertices, corresponding to the three principal axes (X, Y, and Z). The partial order for the X-axis is defined as follows: for two vertices v and w, we say that $v \leq_X w$ if the x-coordinate of v is less than or equal to the x-coordinate of w, and there is an edge between v and w. The partial orders for the Y-axis and Z-axis are defined similarly.

▶ **Definition 2.2.** Two polycubes Q_1 and Q_2 are *order-equivalent* if there exists an isomorphism $f \colon V(Q_1) \to V(Q_2)$ between the quad meshes of Q_1 and Q_2 such that, for all $v, w \in V(Q_1)$ and $\Delta \in \{X, Y, Z\}$, we have that $v \leq_\Delta w$ if and only if $f(v) \leq_\Delta f(w)$.

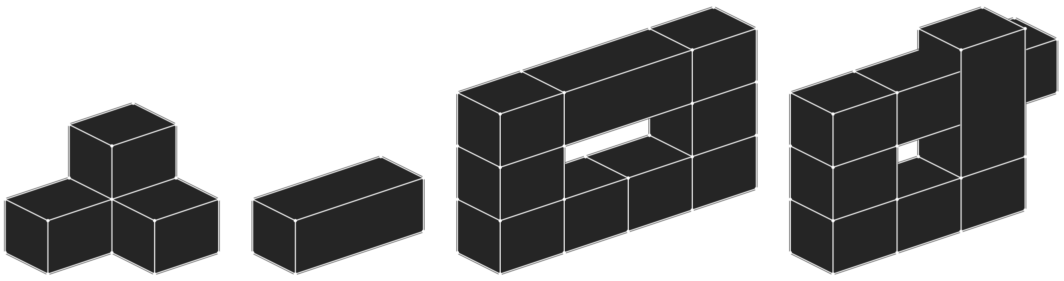


Figure 2 The variety of polycubes that satisfy Definition 2.1.

Our input is a triangulated surface mesh $\mathcal{M}$, which we assume to be an orientable manifold of arbitrary genus bounding a single volume. The mesh $\mathcal{M}$ is embedded in $\mathbb{R}^3$, which implies that each vertex has an associated position in $\mathbb{R}^3$, and each triangular face has a corresponding normal vector.

A polycube segmentation of the surface $\mathcal{M}$ is defined as a partitioning of $\mathcal{M}$ into conforming quadrilateral subsurfaces, called *patches*. Conformity implies that all patch corners coincide exclusively with other patch corners, with no T-junctions. Each patch is assigned a *label* corresponding to one of the six (signed) principal axes: $\{+X, -X, +Y, -Y, +Z, -Z\}$. The partitioning qualifies as a polycube segmentation if there exists a polycube whose faces correspond one-to-one with these patches such that the labels of the patches match the normals of the respective polycube faces, as seen in Figure 1. Quality of the polycube segmentation is hard to define; different qualities may be desirable depending on the downstream usage. In general, the polycube segmentation should be low-distorting, e.g., the mapping between the polycube segmentation and the corresponding polycube should have low mapping distortion.

2.1 Characterization

Polycubes exhibit a dual loop structure, where each loop represents a strip of quadrilateral faces whose center points share a single coordinate [1]; see Figure 3. This structure forms a system of intersecting loops. The places where two loops intersect are *loop intersections*. The parts of a loop bounded by two intersection points (but containing no intersection points) are *loop segments*. The areas bounded by loop segments are *loop regions*. Notice that in the polycube each loop region corresponds to exactly one corner of the polycube.

The loops are labeled as X-, Y-, or Z-loops, depending on the axis perpendicular to the face normals: for example, an X-loop traverses faces with normals aligned to the Y and Z axes. Each loop is oriented by splitting the loop into two (parallel) sides: a *positive* and *negative* side. Crossing a loop from its negative to positive side corresponds to an increase in the associated coordinate, while crossing in the opposite direction results in a decrease. In our figures, we use the colors purple, lighter purple, orange, lighter orange, green, and lighter green for $+X$, $-X$, $+Y$, $-Y$, $+Z$, and $-Z$, respectively.

We can use the full set of X-loops of an oriented loop structure to partition the underlying space (surface or polycube) into regions. We refer to these regions as *X-zones*. Then, the X-graph has a vertex for each X-zone, and a directed edge (u, v) for every X-loop with u and v corresponding to the X-zones on the negative and positive side of the loop, respectively (see Figure 4). We can define the Y-zones, Z-zones, Y-graph and Z-graph similarly.

Given a loop structure where each loop is labeled with either X, Y, or Z, and is oriented with a positive and negative side, there exists a complete set of rules that characterize which of these loop structures correspond to *polycube loop structures*; i.e., the loop structures that are dual to a polycube.

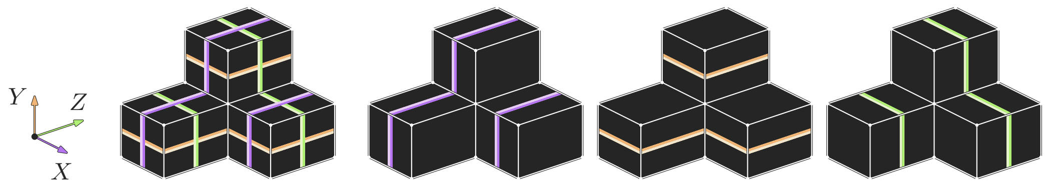


Figure 3 The polycube loop structure consists of the sets X-loops, Y-loops and Z-loops.

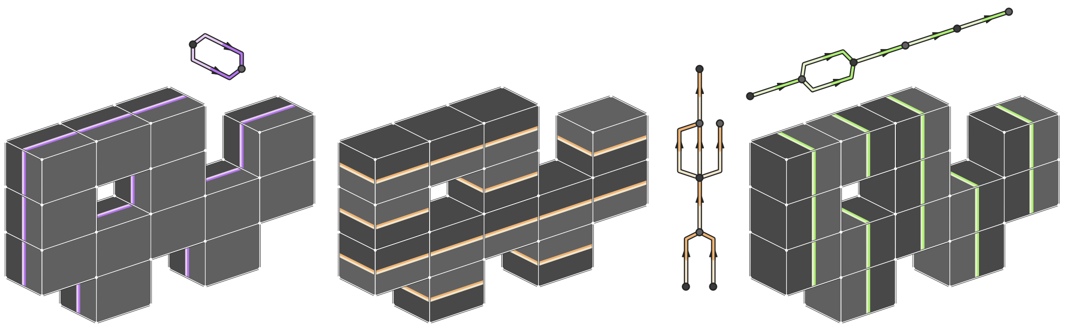


Figure 4 The X-, Y-, and Z-zones along with the X-, Y-, and Z-graph.

▶ **Definition 2.3.** A loop structure is a *polycube loop structure* if:

1. No three loops intersect at a single point.
2. Each loop region is bounded by at least three loop segments.
3. Within each loop region boundary, no two loop segments have the same axis label *and* side label.
4. Each loop region has the topology of a disk.
5. The X-, Y-, and Z-graphs are acyclic.

This means that for every polycube Q there exists a polycube loop structure that forms the dual of Q. And conversely, given a polycube loop structure $\mathcal{L}$, there exists exactly one polycube (up to order-equivalence) that corresponds to $\mathcal{L}$.

3 Algorithm

To compute a polycube segmentation on a surface model $\mathcal{M}$, we propose a method based on the construction and refinement of polycube loop structures. A polycube loop structure provides a complete characterization of polycubes, and polycube segmentations can be seen as the projection of these polycubes onto the surface model. Thus, we can embed a polycube loop structure on a surface model $\mathcal{M}$ and convert it into a polycube segmentation through a primalization step. This involves placing corners within each loop region of the structure and connecting these corners when their corresponding loop regions are adjacent.

As such, the method consists of two main steps: (1) constructing embedded polycube loop structures on a surface model $\mathcal{M}$ and (2) converting these loop structures into polycube segmentations. We can then incorporate these two steps within an iterative framework: we start with a valid polycube loop structure and refine it by adding or removing loops while ensuring its validity, see Figure 5. At each iteration, the current polycube loop structure can be converted into a polycube segmentation for output or intermediate quality evaluation.

The process begins with the initialization of a canonical polycube loop structure based on the genus of the surface model $\mathcal{M}$. For surfaces of genus 0, this structure consists of three interleaving loops aligned with the principal axes (X, Y, and Z) representing a single cube. These three interleaving loops can easily be constructed algorithmically. Surfaces of higher genus require more complex canonical structures. For genus 1, the natural choice is a polycube torus, which can be oriented in three distinct directions, each having a different loop structure. For genus greater than 1, the complexity increases further, as each handle can be oriented independently. Additionally, for a given canonical polycube of higher genus, it remains unclear how to embed its polycube loop structure onto the target surface in a valid manner. As such, we currently initialize solutions for higher-genus surfaces manually.

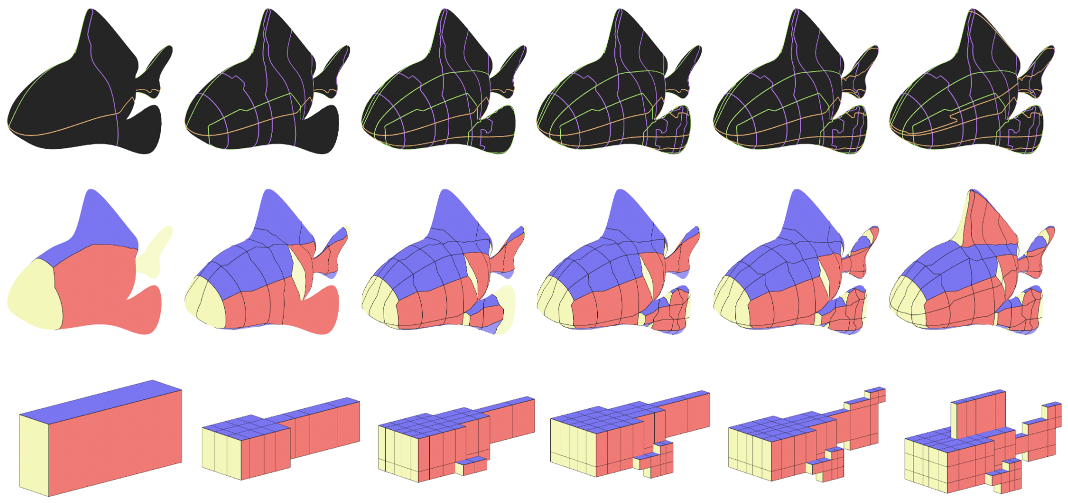


Figure 5 Snapshots of solutions during the iterative construction.

3.1 Dual structure

The polycube loop structure on the surface model $\mathcal{M}$ is represented as a collection of embedded loops. Each loop is a self-closing directed path labeled as either X, Y, or Z. Loop orientation is implicitly defined, with the left side regarded as positive and the right as negative. These loops serve as projections of the polycube's dual loops with a specific label, and as such these loops should approximately align with the corresponding principal axis. To compute valid and suitable loops, we formulate the problem as a constrained shortest path search on $\mathcal{M}$, ensuring validity while using weights to enforce alignment.

Not every loop can be added to the polycube loop structure without violating Definition 2.3. For example, a loop that does not intersect any other loop would violate Condition 2, while a loop forming a region bounded by seven segments would violate Condition 3. To address this, we use the method from [5] to enumerate all valid intersection patterns that correspond to unique polycube loop structures. These intersection patterns serve as hard constraints, guiding the path-finding algorithm on where the loops should intersect.

To find well-aligned loops, we define the following property. Let p be a point on a loop, $d(p)$ the vector tangent to the loop at p in the direction of traversal, and $n(p)$ the normal vector at p with respect to the surface model $\mathcal{M}$. For a well-aligned X-loop we want the cross product $d(p) \times n(p)$, that is the vector perpendicular to $d(p)$ and $n(p)$, to be aligned with the X-axis for all points on the loop. The same principle applies to Y-loops and Z-loops.

We obtain this property by assigning suitable weights to edges on the surface model $\mathcal{M}$. Each edge e on $\mathcal{M}$ has a defined direction vector $d(e)$ and normal vector $n(e)$. When computing the path of an X-loop, we assign the weight $w_X(e) = \text{angle}(d(e) \times n(e), \overrightarrow{X})^\alpha$ to each edge e by measuring its alignment with respect to $\overrightarrow{X}$, the direction of the X-axis. The strictness factor α increases the penalty for misalignment. Analogously, we can compute well-aligned Y-loops and Z-loops using different weights w_Y and w_Z computed in a similar manner. Note that the edges are directed, and the weights clearly favor a specific direction. As a result, the computed loops will also be consistently oriented.

A well-aligned loop can then be computed as a (non-empty) shortest path from a point p on $\mathcal{M}$ to itself, using the constraints of the intersection patterns defined earlier and the appropriate weights for alignment. The resulting loops are both valid and well-aligned.

3.2 Primalization

To convert the polycube loop structure to a polycube segmentation, we perform primalization of the polycube loop structure. A patch corner is placed within each loop region and adjacent patch corners are connected by non-intersecting paths. Since the polycube loop structure is the dual to a polycube segmentation, the primalization step results in a valid solution.

Two considerations guide the placement of corners. First, corners must be placed in regions consistent with their specific corner types (see [5] for all possible types). Second, corners must align globally: In a polycube, there are typically many corners that share either their x-, y-, or z-coordinate. To preserve these properties, the polycube segmentation must exhibit similar alignment and consistency across the surface.

4 Results

We have implemented this approach within an iterative framework, embedding it into an evolutionary algorithm that optimizes polycube segmentations by evaluating solutions based on three key metrics. First, regularity is maximized by minimizing the number of corners with a degree other than 4. Second, alignment is maximized by ensuring that the surface normal at each point on $\mathcal{M}$ closely corresponds to the label implied by the polycube segmentation. Finally, orthogonality is maximized by aiming for near 90-degree corners in each patch of the polycube segmentation. A showcase of our results can be found in Figure 6.

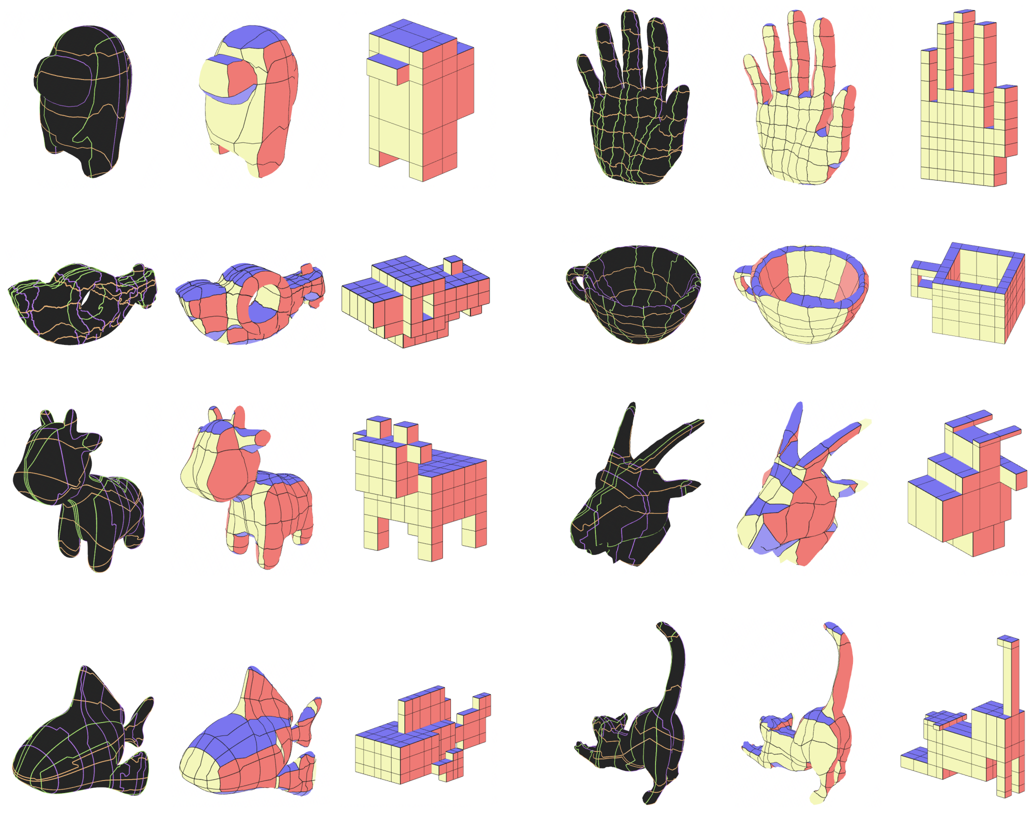


Figure 6 The polycube loop structure, and its corresponding polycube segmentation and polycube for a variety of input surface models.

References

1 Therese Biedl and Burkay Genc. When can a graph form an orthogonal polyhedron? In *Proceedings of the 16th Canadian Conference on Computational Geometry (CCCG)*, 2004. URL: `http://www.cccg.ca/proceedings/2004/15.pdf`.

2 Corentin Dumery, François Protais, Sébastien Mestrallet, Christophe Bourcier, and Franck Ledoux. Evocube: A genetic labelling framework for polycube-maps. *Computer Graphics Forum*, 41(6), 2022. `doi:10.1111/cgf.14649`.

3 Marco Livesu, Nicholas Vining, Alla Sheffer, James Gregson, and Riccardo Scateni. Polycut: monotone graph-cuts for polycube base-complex construction. *ACM Transactions on Graphics*, 32(6), 2013. `doi:10.1145/2508363.2508388`.

4 Nico Pietroni, Marcel Campen, Alla Sheffer, Gianmarco Cherchi, David Bommes, Xifeng Gao, Riccardo Scateni, Franck Ledoux, Jean Remacle, and Marco Livesu. Hex-mesh generation and processing: A survey. *ACM Transactions on Graphics*, 42(2), 2022. `doi:10.1145/3554920`.

5 Maxim Snoep, Bettina Speckmann, and Kevin Verbeek. Polycubes via dual loops. In *Proceedings of the 2025 SIAM International Meshing Roundtable (IMR)*, 2025. To appear. URL: `https://arxiv.org/abs/2410.16865`.

6 Dmitry Sokolov and Nicolas Ray. Fixing normal constraints for generation of polycubes. research report, LORIA, 2015. URL: `https://inria.hal.science/hal-01211408`.

7 Marco Tarini, Kai Hormann, Paolo Cignoni, and Claudio Montani. Polycube-maps. *ACM Transactions on Graphics*, 23(3), 2004. `doi:10.1145/1015706.1015810`.

8 Hongyu Wang, Ying He, Xin Li, Xianfeng Gu, and Hong Qin. Polycube splines. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling (SPM)*, 2007. `doi:10.1145/1236246.1236281`.

Optimal covering of rectangular grid graphs with tours of constrained length*

Sergey Bereg[1], **Jesús Capitán**[2], **José-Miguel Díaz-Bañez**[3], **José-Manuel Higes-López**[3], **Miguel-Angel Pérez-Cutiño**[3,4], **Vanesa Sánchez-Canales**[3], **and Inmaculada Ventura**[3]

1 **Department of Computer Science, University of Texas at Dallas, USA**
besp@utdallas.edu

2 **Multi-robot & Control Systems group, University of Seville, Spain**
jcapitan@us.es

3 **Department of Applied Mathematics, University of Seville, Spain**
dbanez@us.es
jhiges@us.es
vscanales@us.es
iventura@us.es

4 **Virtualmechanics S.L, Seville, Spain**
m.perez@virtualmech.com

Abstract

Given a rectangular grid graph, we study the problem of covering the entire graph with tours that start and end at a given corner and whose lengths do not exceed a given threshold, while minimizing a quality measure. We consider two objective functions: minimizing the number of tours, and minimizing the total length of the tours. We present an algorithm that computes the optimal solution for both objectives in linear time relative to the size of the grid.

1 Introduction

Let G be a rectangular grid graph, that is, a grid graph defined by an $n_c \times n_r$ rectangle such that all its interior faces are unit squares (i.e., the graph contains no holes). Without loss of generality, we assume that the number n_c of columns is less than or equal to the number n_r of rows in G. Given a special vertex or base station $\mathcal{B}$ located in the lower left corner of G, and a number $L \geq 2(n_r + n_c - 2)$, we aim to compute a set of tours that covers the entire grid. We consider that the length of each tour is constrained by L, and they must start from and return to $\mathcal{B}$. Since the length of any tour in G is even, we assume that L is even. Our objective is to solve two independent minimization problems, given by the following measures: (1) the number of tours, and (2) their total length, which is equivalent to minimize the total number of repeats of the tours, that is, the total number of times each vertex is traversed by all tours except its first traversal. Each problem is identified as **Min-Tours Problem (MTP)** and **Min-Repeats Problem (MRP)**, respectively.

Our case study is inspired by solar power plants, as their topologies suggest to consider discretizations to grid points, and they are typically monitored by drones that move according to the Manhattan distance; see Figure 1. Our problems are related to the family of Vehicle

* Supported in part by grants PID2020-114154RB-I00, TED2021-129182B-I00, and DIN2020-011317 funded by MCIN/AEI/10.13039/501100011033 and the European Union NextGenerationEU/PRTR. The problems studied here were partially solved during the First Soldrone Workshop, June 2023, Barbate, Spain. The authors would like to thank other participants for helpful comments.

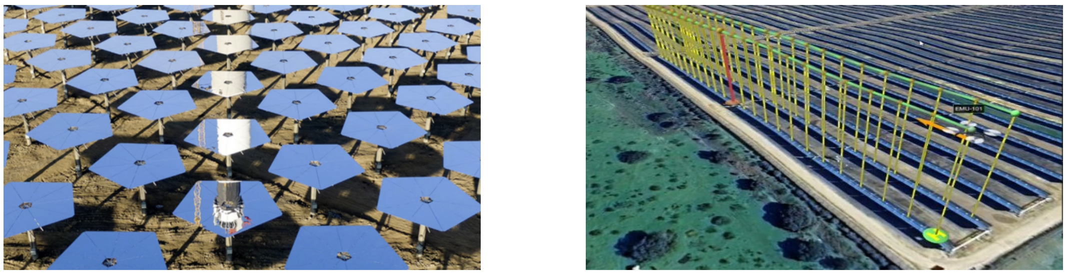


Figure 1 Stellio Heliostat solar plant (left), and Drone route planning based on discrete points in Parabolic Trough solar plants (right).

Routing Problems (VRP) [6] that fall within the framework of the TSP-type problems. The Traveling Salesman Problem (TSP) on solid grid graphs, that is, without holes, is one of the long-standing open problems in the prominent list *The Open Problems Project*[1] (TOPP). As a consequence, finding a set of length-constrained tours of minimal total length covering a graph remains open in general solid grids. One of the questions addressed by our paper is whether that problem is polynomially solvable in rectangular grid graphs for the case in which all tours start in a corner. Reducing complex problems to grids has been extensively proposed in the existing literature [5, 7, 2, 1, 8, 3, 4].

An interesting result from our research is the connection between MTP and MRP. This connection is achieved by solving the following auxiliary problem termed as **Range Level Problem (RLP)**, which holds independent theoretical interest: *Define the level $i \in \mathbb{N}$ of G as the set of vertices in G at Manhattan distance i from $\mathcal{B}$. Determine the maximum level i such that for every set T of tours covering G with minimum number of repeats, each tour in T contains at least one point above the level i.*

Contributions. Let n be the total number of vertices of G, and $k_{\min}$ the minimum number of tours required to cover the grid G. Our main contributions can be summarized as follows: (1) $k_{\min}$ can be found in constant time. (2) The solution to RLP is bounded below by $2k_{\min} - 3$, which implies that the total number of repeats of any set of tours covering G is at least $2k_{\min}^2 - 2k_{\min} - 1$. (3) In $O(n)$ time we can obtain a set T of tours covering G such that $|T| = k_{\min}$, and the number of repetitions of T in G is minimized.

2 The Algorithms

In this section, we introduce basic definitions and outline the general approach of our algorithms. Our method consists of subdividing the rectangular grid graph into disjoint regions, followed by an iterative construction of walks that are later transformed into tours. At first glance, one might assume that a simple greedy strategy suffices to cover G. However, ensuring optimality is nontrivial, even in structured settings such as rectangular grid graphs. For instance, a straightforward left-to-right sweeping algorithm fails to achieve an optimal solution; see the left example in Figure 2. This highlights the need of carefully designing an optimal procedure. We begin by introducing some additional notation.

▶ **Definition 2.1.** The level i of G is the set of vertices in G that are at the same Manhattan distance i from $\mathcal{B}$. Note that at level i there are $i + 1$ vertices if $i \leq n_c - 1$.

[1] `http://cs.smith.edu/~orourke/TOPP/`

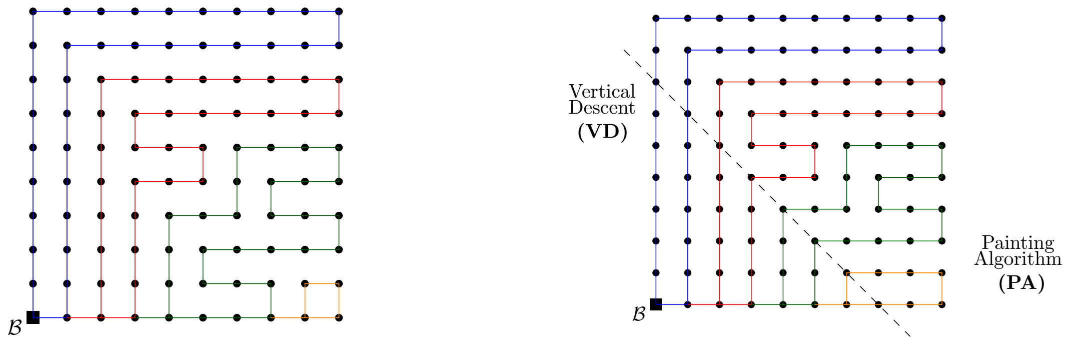


Figure 2 Overview of the covering strategy for a rectangular grid of 10×10, using tours with length at most 36. The example to the left shows a greedy strategy with total length 128. To the right, the covering obtained from our algorithms with a total length of 124. The dashed line represents the level from which each algorithm is applied.

▶ **Definition 2.2.** Let A_i denote the solid grid graph obtained by removing from G all vertices located below level i. Here, level i of G is referred to as the baseline of A_i. Vertices at the baseline of A_i are denoted from left to right with $t_1, t_2, \dots, t_{i+1}$.

Our strategy divides the grid into two parts: A_{2r-1} and $G \setminus A_{2r-1}$ for every $r \in \{1, \dots, \lceil \frac{n_c}{2} \rceil\}$. First, we focus on computing a set of r walks covering A_{2r-1} such that the walks in this set are pairwise disjoint. Moreover, each walk w will be *as long as possible* while meeting some constraints, that is, the portion not covered by w is always a grid in the family depicted in Figure 3. Then, each walk in A_{2r-1} will be transformed into a tour starting and ending at $\mathcal{B}$ by vertically descending from the baseline of A_{2r-1} to the bottom row of G, and then using a horizontal line to connect with $\mathcal{B}$, if needed. We denote this last process as **Vertical Descent (VD)**. This combined strategy completely covers the grid; see right example in Figure 2. If the length l of the walks defined in A_{2r-1} are upper bounded by $L - 2(2r-1)$, then the resulting tours meet the length constraint.

▶ **Definition 2.3.** Let $\mathcal{S}$ be the family of grid graphs represented in Figure 3, and defined as follows:

- Any rectangular grid is in $\mathcal{S}$. A rectangular grid with a columns and b rows is denoted by $\mathcal{R}(a, b)$; see left drawing in Figure 3.
- $S_1(a, b, c)$ with $a > c$ is a grid graph in $\mathcal{S}$ obtained from a rectangular grid graph $\mathcal{R}(a, b)$ by removing its subgraph $\mathcal{R}(a - c, 2)$ from the upper left corner; see central drawing in Figure 3.
- $S_2(a, b)$ is a grid graph in $\mathcal{S}$ obtained from $\mathcal{R}(a, b-2)$, $a, b \geq 3$, by removing its subgraph $\mathcal{R}(1, 2)$ from the upper left corner and adding a graph $\mathcal{R}(2, 2)$ right aligned above its right top corner; see right drawing in Figure 3.

When making a reference to a particular subfamily of the grids contained in $\mathcal{S}$, we use the expression $\mathcal{R}(\cdot)$, $S_1(\cdot)$, or $S_2(\cdot)$ for convenience. Similarly, the expression $S_1(\cdot, \cdot, 1)$ will denote any grid in the subfamily of $S_1(\cdot)$ with $c = 1$.

▶ **Definition 2.4.** For any grid graph H, let z be the number of vertices in the bottom row of H. When $z > 1$ we define a new grid graph $AddRow(H)$ as the grid graph resulting from adding one row with z vertices at the bottom of H, followed by the removal of the left-most vertex of this newly added row.

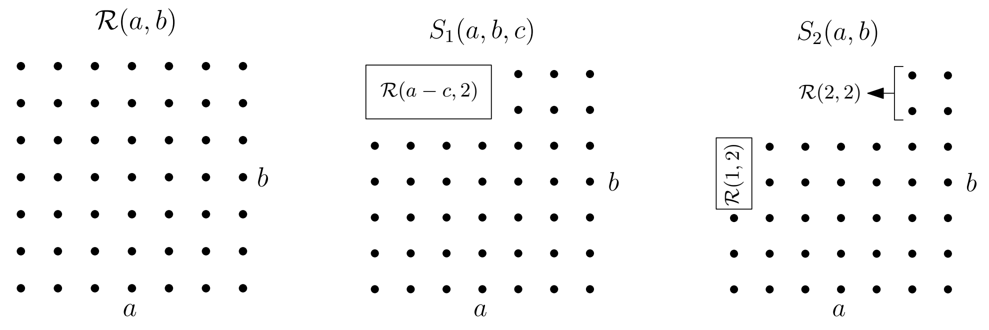


Figure 3 General grids defined by family $\mathcal{S}$.

▶ **Definition 2.5.** Given $U \in \mathcal{S}$ with $a > 1$ columns, and an integer l being at least the perimeter of U, let $A = \mathit{AddRow}(U)$ and p be a walk over A such that its length $\ell(p) \leq l$. We define $A(p)$ as the portion of A that is covered by p, and R as the complement, i.e., $R = A \setminus A(p)$. In addition, we say p is $\mathcal{S}$-maximal if all of the following conditions hold.

1. $R \in \mathcal{S} \setminus S_1(\cdot,\cdot,1)$.
2. Assuming that vertices in a row (column) of A are ordered from left to right (top to bottom), for any row (column) there exists a vertex v (s) such that all vertices v' (s') of such row (column) with $v' \leq v$ ($s' \leq s$) belongs to $A(p)$, and all the remaining vertices in such row (column), if any, are in R.
3. Either $\ell(p) = l$, or $R \in \{\emptyset, \mathcal{R}(a-2,1), S_1(a-2,3,2)\}$.

We are now ready to describe the Painting Algorithm (**PA**). **PA** divides A_{2r-1} into two portions, as shown in Figure 4(b): 1) a zone $U_1 \in \mathcal{S}$ containing the row of t_1 and all the rows above; 2) a trapezoid δ_1 containing $A_{2r-1} \setminus \mathit{AddRow}(U_1)$. At step $i > 1$ of the algorithm, two vertices (t_{2i-1} and t_{2i}) in the baseline of A_{2r-1} will be connected with an $\mathcal{S}$-maximal walk. In addition, $U_i \in \mathcal{S}$ will be the union of the uncovered vertices of U_{i-1} and the vertices in the row that contains t_{2i-1}, and δ_i will be a trapezoid containing $A_{2r-1} \setminus \bigcup_{1 \leq j \leq i} \mathit{AddRow}(U_j)$; see Figure 4 for a complete example. The final step of **PA** finalizes the covering depending on the parity of $|V(G)|$; see Figures 5 and 6. We use **PA** to prove the following theorems, but the proof and the formal definition of **PA** are omitted in this version.

▶ **Theorem 2.6.** *Let P be a set with r walks of length at most l starting and ending at the baseline of A_{2r-1} such that P covers A_{2r-1}. Then, there exists a set P' of pairwise disjoint walks with length at most l starting and ending at the baseline of A_{2r-1} such that*

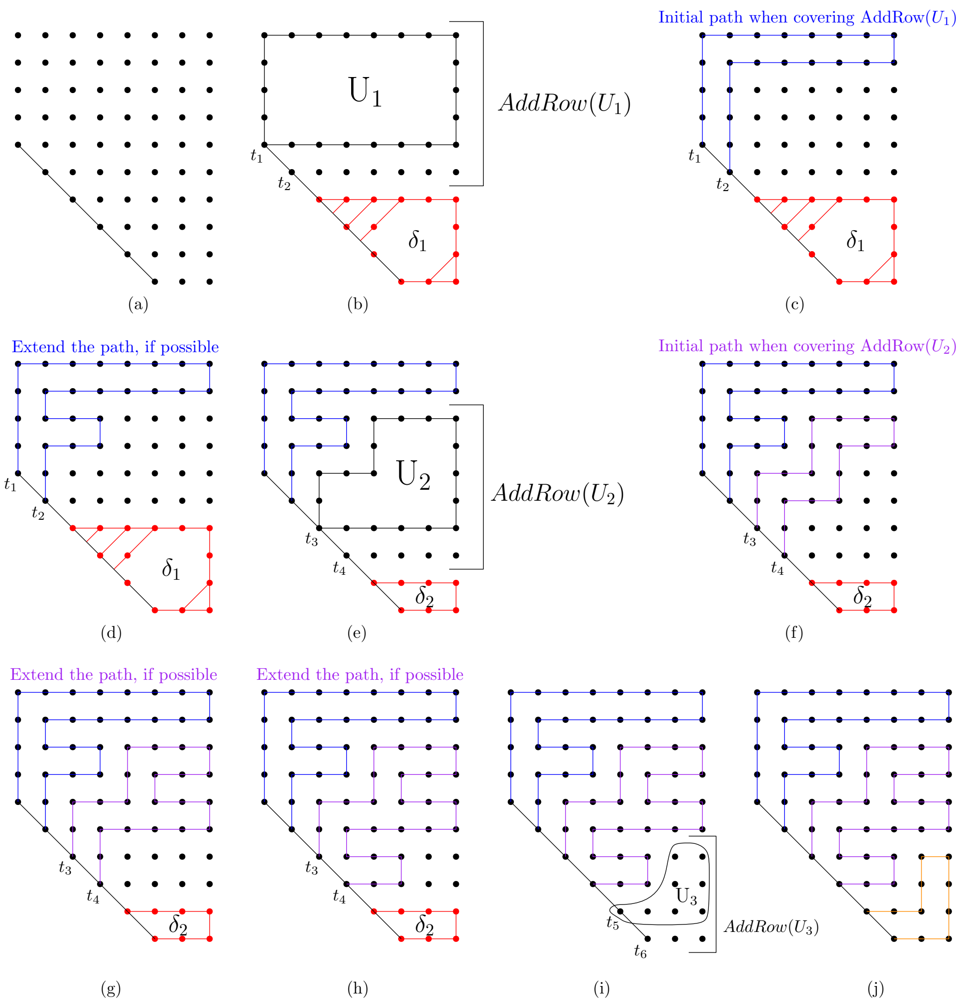
(i) P' covers A_{2r-1},
(ii) at most one walk in P' contains a single repetition, and
(iii) the ending and starting vertices of all walks in P' are different.

▶ **Theorem 2.7.** ***PA** is optimal for covering A_{2r-1} with r walks and minimum number of repeats. Moreover, it runs in $O(|V(A_{2r-1})|)$ time.*

3 Min-Tours Problem

We begin with a useful result:

▶ **Lemma 3.1.** *Let P be any set of r walks covering A_{2r-1}, with each walk starting and ending on its baseline. If $|V(G)|$ is odd, then there is at least one repetition of P in A_{2r-1}.*



Initial path when covering AddRow(U_1)
U_1
$AddRow(U_1)$
t_1
t_2
δ_1
(a)
(b)
(c)
Extend the path, if possible
U_2
$AddRow(U_2)$
t_3
t_4
δ_2
(d)
(e)
Initial path when covering AddRow(U_2)
(f)
Extend the path, if possible
Extend the path, if possible
U_3
$AddRow(U_3)$
t_5
t_6
(g)
(h)
(i)
(j)

Figure 4 Example of Painting Algorithm for covering A_{2r-1} with $l = 26$. (a) A_5 with 10 rows and 8 columns; (b) initial zone division in the algorithm; (c)-(d) creating the first walk iteratively; (e) first walk (in blue) and second zone division; (f)-(g)-(h) creating the second path iteratively; (i) first two walks (in blue) and third zone division; (j) final covering.

Proof. We will prove that if the number of repetitions of P in A_{2r-1} is 0, then $|V(G)|$ is even. Since walks in P start and end on the baseline of A_{2r-1}, they are disjoint if and only if the following condition is met: for any walk starting in point i and ending in point $j > i$ of the baseline it follows that any walk starting at $i < z < j$ ends at $z < w < j$. Thus, P can be easily transformed into disjoint cycles in G, which can be joined to create a Hamiltonian cycle. Therefore, $|V(G)|$ is even and the lemma holds. ◀

Since **PA** is optimal for covering A_{2r-1}, the following decision question is solvable in constant time: *can A_{2r-1} be covered with r walks of length at most $L - 2(2r-1)$?* A_{2r-1} can be covered with 0 repeats iff $|V(G)|$ is even; see Lemma 3.1 and Theorems 2.6 and 2.7. Then, if $k_{\min} > 1$ we can establish the following formula to obtain its value:

$$k_{\min} = \begin{cases} \min\{r : |V(A_{2r-1})| \le r(L - 2(2r-1) + 1)\} & \text{if } |V(G)| \text{ is even,} \\ \min\{r : |V(A_{2r-1})| + 1 \le r(L - 2(2r-1) + 1)\} & \text{if } |V(G)| \text{ is odd.} \end{cases} \tag{1}$$

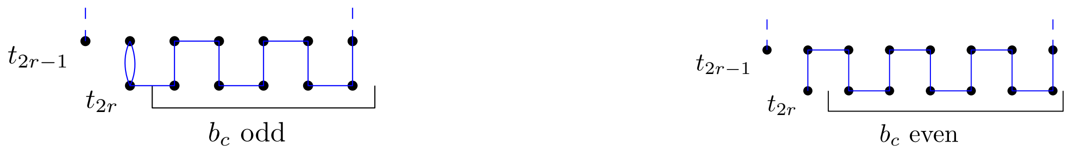


Figure 5 Final step in **PA** when the grid graph has an odd number of rows. One repetitions is obtained when the number of columns is odd (left), and no repetitions are obtained for the even case (right).

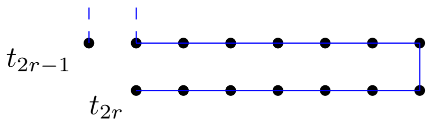


Figure 6 Final step in the Painting Algorithm when A_{2r-1} has an even number of rows.

Solving the above equation, we get the following theorem:

▶ **Theorem 3.2.** *Give n_c, n_r and L, the minimum number of tours of length at most L required to cover a rectangular grid graph with n_c columns and n_r rows can be obtained with the formula*

$$k_{min} = \begin{cases} \left\lceil \frac{L+2-\sqrt{-8n_c n_r + L^2 + 4L + 4}}{4} \right\rceil, & n_c n_r \textit{ is even} \\ \left\lceil \frac{L+2-\sqrt{-8n_c n_r + L^2 + 4L - 4}}{4} \right\rceil, & n_c n_r \textit{ is odd.} \end{cases} \tag{2}$$

4 Min-Repeats Problem

As $k_{\min}$ can be obtained in constant time, we can achieve a set of tours covering G using **PA** and **VD** over $A_{2k_{\min}-1}$; for simplicity we use the notation $\mathbf{PA}_{k_{\min}}$**+VD**. In this section, we start by analyzing the conditions for which $\mathbf{PA}_{k_{\min}}$**+VD** is optimal for solving the Min-Repeats Problem. Then, we extend **PA** to consider two additional cases by modifying its final walk, which completes all possible scenarios. We start by providing a lower bound to the Range Level Problem using the following structural results.

▶ **Lemma 4.1.** *Let T be a set with k_{min} tours covering G. Then every tour in T contains at least one point above the level $2k_{min} - 3$.*

Using the Pigeonhole Principle and Lemma 4.1, we get:

▶ **Lemma 4.2.** *For any set of k_{min} tours covering G, the minimum number of repeats is lower bounded by $2k_{min}^2 - 2k_{min} - 1$, and upper bounded by $2k_{min}^2 - 2k_{min} + 1$.*

▶ **Lemma 4.3.** *Let T be an optimal solution to **MRP**. Then $|T| = k_{min}$.*

From Lemma 4.3 and Lemma 4.1, a lower bound to RLP can be readily obtained:

▶ **Theorem 4.4.** *The solution z to RLP satisfies $2k_{min} - 3 \leq z$.*

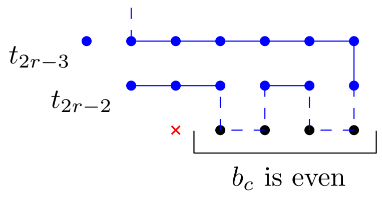


Figure 7 Final step in the modified Painting Algorithm when $|V(G)|$ is odd. The red cross indicates the point in the baseline of A_{2r-2} that will not be covered.

4.1 Optimality conditions for $\textbf{PA}_{k_{\textsf{min}}}$ + VD when $k_{\textsf{min}} < \lceil \frac{n_c}{2} \rceil$

In what follows, we distinguish the cases in which $\textbf{PA}_{k_{\text{min}}}$ + **VD** obtains a covering of G with minimum number of repeats. These cases are defined by the following two theorems.

▶ **Theorem 4.5.** *If there exists a set of tours T covering G with minimum number of repeats such that every tour in T traverses the level $2k_{min} - 1$, then $\textbf{PA}_{k_{min}}$ + **VD** produces an optimal solution for Min-Repeats Problem.*

▶ **Theorem 4.6.** *If $|V(G)|$ is even, then $\textbf{PA}_{k_{min}}$+**VD** obtains a set of tours covering G with minimum number of repetitions.*

For demonstrating the previous theorems we use Lemmas 4.2 and 4.3, and the following lemma which also provides the ingredient describing the case where $\textbf{PA}_{k_{\text{min}}}$ + **VD** is not optimal when $k_{\text{min}} < \lceil \frac{n_c}{2} \rceil$.

▶ **Lemma 4.7.** *Let d be the number of tours in an optimal solution T to the Min-Repeats Problem that does not reach the level $2k_{min} - 1$. If $d > 0$, then $d = 1$.*

4.2 Additional cases

Case 1: $k_{\textsf{min}} < \lceil \frac{n_c}{2} \rceil$, $|V(G)|$ is odd and one tour is *below* the level $2k_{\textsf{min}} - 1$. We describe a modified version of **PA** for this scenario, termed as **PAO**. For a given r, **PAO** aims to compute a covering of A_{2r-2} with $r-1$ paths and one point. This point will be the last point on the baseline of A_{2r-2}; see the red cross in Figure 7. Moreover, the $r-1$ walk obtained with **PAO** will end as indicated in the figure. As **PAO** is only applied when $|V(G)|$ is odd, the case described in Figure 7 is complete. Moreover, **PAO** covers $A_{2k_{\text{min}}-2}$ without repetitions; hence the following holds.

▶ **Theorem 4.8.** *If $|V(G)|$ is odd and there exists a set T of tours covering G with minimum number of repeats such that one tour in T does not traverse the level $2k_{min} - 1$, then **PAO+VD** produces a covering of G with minimum number of repeats in $O(|V(G)|)$ time.*

Case 2: $k_{\textsf{min}} = \lceil \frac{n_c}{2} \rceil$. If n_c is even, clearly $\textbf{PA}_{k_{\text{min}}}$+**VD** is optimal. If n_c is odd, we can demonstrate that the minimum number of repeats for covering A_{n_c-1} is half of the length of the smallest path in an optimal solution to the Min-Repeats Problem; hence we focus on minimizing the length of the smallest path. To this end, we change the last walk of **PA** to cover a grid of the form $\mathcal{R}(1, b)$, which is basically a line. We term this version as **PAR**, and the idea to demonstrate that **PAR** is optimal when n_c is odd is to first demonstrate that the initial $k_{\text{min}} - 1$ walks of **PAR** are of maximum length. Then we have:

▶ **Theorem 4.9.** *Consider $k_{min} = \lceil \frac{n_c}{2} \rceil$. An optimal solution to the Min-Repeats Problem can be obtained with $\textbf{PA}_{k_{min}}$+**VD** when n_c is even, or with **PAR+VD** when n_c is odd.*

5 Conclusion

Our strategy can be extended to cases where the base station is located at any vertex on the boundary of a rectangular grid graph. Our intention is to generalize the concept of levels from lines to triangular regions. The problem remains also open for other solid grid graphs.

References

1 Esther M Arkin, Sándor P Fekete, Kamrul Islam, Henk Meijer, Joseph SB Mitchell, Yurai Núñez-Rodríguez, Valentin Polishchuk, David Rappaport, and Henry Xiao. Not being (super) thin or solid is hard: A study of grid hamiltonicity. Computational Geometry, 42(6-7):582–605, 2009.

2 Esther M Arkin, Sándor P Fekete, and Joseph SB Mitchell. Approximation algorithms for lawn mowing and milling. Computational Geometry, 17(1-2):25–50, 2000.

3 Fatemeh Keshavarz-Kohjerdi and Alireza Bagheri. A linear-time algorithm for finding hamiltonian (s,t)-paths in even-sized rectangular grid graphs with a rectangular hole. Theoretical Computer Science, 690:26–58, 2017.

4 Mathieu Mari, Anish Mukherjee, Michał Pilipczuk, and Piotr Sankowski. Shortest disjoint paths on a grid. In Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 346–365. SIAM, 2024.

5 Wen Lea Pearn. Solvable cases of the k-person Chinese postman problem. Operations Research Letters, 16(4):241–244, 1994.

6 Paolo Toth and Daniele Vigo. Vehicle routing: problems, methods, and applications. SIAM, 2014.

7 Christopher Umans and William Lenhart. Hamiltonian cycles in solid grid graphs. In Proceedings 38th Annual Symposium on Foundations of Computer Science, pages 496–505. IEEE, 1997.

8 Arthur van Goethem, Irina Kostitsyna, Marc van Kreveld, Wouter Meulemans, Max Sondag, and Jules Wulms. The painter's problem: covering a grid with colored connected polygons. In International Symposium on Graph Drawing and Network Visualization, pages 492–505. Springer, 2017.

On the Spectrum from Convex to Connected r-gather on Graphs[*]

Thomas C. van Dijk[1], Wolf Kißler[2], Lukas Plätz[2], Marena Richter[3], and Jonathan Rollin[4]

1 **TU Eindhoven, Department of Mathematics & Computer Science**
2 **Ruhr-Universität Bochum, Faculty for Computer Science**
3 **Universität Bonn, Institute of Computer Science**
4 **FernUniversität in Hagen, Faculty of Mathematics and Computer Science**

Abstract

In this paper we explore several notions of convexity in the setting of r-gather, establish the hierarchy among the variants, and prove their hardness on general graphs. We give an exact polynomial time algorithm for convex r-gather on unweighted trees and a 4-approximation algorithm for connected r-gather on general weighted graphs.

1 Introduction

In the r-gathering problem, we are given a set of nodes and centers in a metric space and create a clustering by assigning nodes to selected centers while minimizing the maximum distance from the nodes to their assigned center, where at least r nodes must be assigned to each selected center. The variant where the centers are not given in advance but can be chosen arbitrarily is called r-gather. It was introduced to transform the problem of anonymizing sensitive data into a clustering problem [1]. Both problems are NP-hard in general, and optimal approximation algorithms are known [1, 2]. Both problems are also hard on weighted spider graphs [7] and r-gathering has efficiently solvable variants on trees [6].

Besides minimizing the distances of nodes in the clusters, further connectivity or proximity properties might be desirable. For example, r-gatherings where each node is assigned to the nearest selected center were studied [2]. Many notions of connectivity and other clustering constraints are known under the general term of facility location problems [5]. We want to highlight the notion of geodesic convexity in graphs. In the literature, the standard definition of a convexity is that for each pair of nodes in the same cluster all shortest paths are required to lie within that cluster. We use a weaker version in which we only require that at least one shortest path lies within the cluster. That is partly motivated by the application of the Anonymous Routing Problem [4]. Pelayo surveyed many results [8], including that partitioning general graphs into a given number of convex clusters is NP-hard [3].

Our Contribution In Section 2, we formulate and compare multiple variants of r-gather on graphs that yield a spectrum from convexity to connectivity. In Section 3, we introduce an algorithm that computes a convex r-gather with optimal diameter on unweighted trees with n nodes in time $\mathcal{O}(nr^3 \log r)$. We end with Section 4, where we give a 4-approximation algorithm for the optimal diameter of connected r-gather on general weighted graphs.

[*] This work was started at the Rhein-Ruhr Computational Geometry Workshop 2023.

2 Relations between Variations of r-Gather

We consider weighted undirected graphs $G = (V, E)$. Given a subset (cluster) $V' \subseteq V$ and nodes v,$w \in V'$, we denote by $\mathrm{d}_{G[V']}(v, w)$ the length of the shortest v-w-path in the induced graph $G[V']$. If no such path exists, we set $\mathrm{d}_{G[V']}(v, w) = \infty$. An optimum solution to the *r-gather* problem, is a partition of V into subsets V_1, $V_2, \ldots$ each of size at least r such that $\max_i \max_{v,w \in V_i} \mathrm{d}_G(v, w)$ is minimized.[1] We assume $|V| \geq r$, to ensure the existence of solutions to r-gather, and $r > 1$, as otherwise, singleton sets yield a clustering with diameter 0, which is optimal. The following secondary conditions define the variations:

- *(unrestricted) r-gather* has no further conditions,
- *connected r-gather* requires that for each i the induced subgraph $G[V_i]$ is connected,
- *convex r-gather* requires that $\forall i \forall v, w \in V_i \colon \mathrm{d}_G(v, w) = \mathrm{d}_{G[V_i]}(v, w)$,
- for $\alpha \geq 1$, *α-sum-convex r-gather* requires that $\forall i \colon \sum\limits_{v,w \in V_i} \mathrm{d}_{G[V_i]}(v, w) \leq \alpha \sum\limits_{v,w \in V_i} \mathrm{d}_G(v, w)$,
- for $\alpha \geq 1$, *α-ratio-convex r-gather* requires that $\forall i \forall v, w \in V_i \colon \mathrm{d}_{G[V_i]}(v, w) \leq \alpha \mathrm{d}_G(v, w)$.

Denote the respective optimal solutions by d^*, d^*_{conn}, d^*_{conv}, $d^*_{\alpha-\text{sum}}$, $d^*_{\alpha-\text{ratio}}$. We omit G and r in the notation as they are usually clear from the context. Observe that if some connected component of G has less than r nodes, then connected r-gather has no solution and d^*, d^*_{conv}, $d^*_{\alpha-\text{sum}}$, $d^*_{\alpha-\text{ratio}} = \infty$ (as $V_1 = V$ is a solution for these variants but at least one cluster spans multiple components in each solution). If each connected component has at least r nodes, then the connected components of G form a partition of V that is a solution for each variant of r-gather and, hence, all corresponding values are finite. Additionally, for each variant, the optimal solution has no clusters that span multiple components and a globally optimal solution can be obtained by solving r-gather on each component separately.

Throughout this section, we present proof sketches and visual explanations. Detailed proofs will be included in the full version of this paper.

The following lemma shows some basic relations between the variants.

▶ **Lemma 2.1.** *For each $\alpha \geq 1$ and each graph, $d^* \leq d^*_{\text{conn}} \leq d^*_{\alpha-\text{sum}} \leq d^*_{\alpha-\text{ratio}} \leq d^*_{\text{conv}}$ holds. Moreover, for $\alpha = 1$, we have $d^*_{\alpha-\text{sum}} = d^*_{\alpha-\text{ratio}} = d^*_{\text{conv}}$.*

Proof sketch. We may assume that G is connected, since otherwise, the arguments would hold for each connected component individually. Any convex solution is α-ratio-convex for each $\alpha \geq 1$ and any α-ratio-convex solution is α-sum-convex. These three notions coincide for $\alpha = 1$. If $d^*_{\alpha-\text{sum}}$ is finite, any α-sum-convex solution is connected. Lastly, any connected solution is also an unrestricted solution. ◀

We have the following general upper bound for connected r-gather on unweighted graphs.

▶ **Lemma 2.2.** *For any unweighted graph, it holds that $d^*_{\text{conn}} \leq 2r - 2$.*

On weighted trees, all restricted variants coincide. However, the following lemma shows that there are cases where the connected solution is not better than a 2-approximation to the unrestricted r-gather even if G is an unweighted spider, i.e., a tree where at most one node has degree larger than 2 (see Figure 1).

▶ **Lemma 2.3.** *For any $\varepsilon > 0$, there exists an unweighted spider graph and a minimum cluster size r such that $d^*_{\text{conn}} > (2 - \varepsilon)d^*$.*

[1] The literature commonly considers center clustering and minimizes the radius instead.

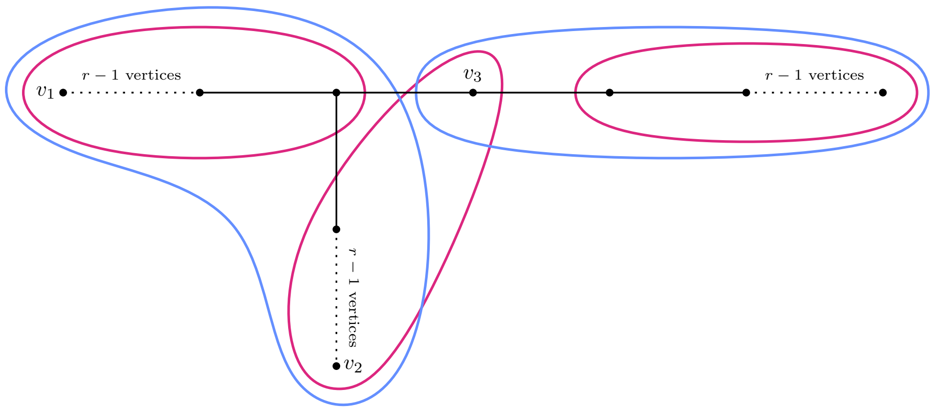


Figure 1 Visual proof of Lemma 2.3: Optimal connected(blue)/unrestricted(red) solutions.

In contrast to trees, we show lower bounds for approximating d^*_{conn} in terms of d^*_{conv} or $d^*_{\alpha-\text{ratio}}$ on weighted planar graphs. Other than the results in Lemma 2.3, these bounds hold for constant r and even for a constant size of G (see Figures 2 and 3).

▶ **Lemma 2.4.** *For all $\varepsilon > 0$, $r = 3$, there is a planar graph G such that $d^*_{\text{conv}} \geq (\frac{5}{2} - \varepsilon)d^*_{\text{conn}}$.*

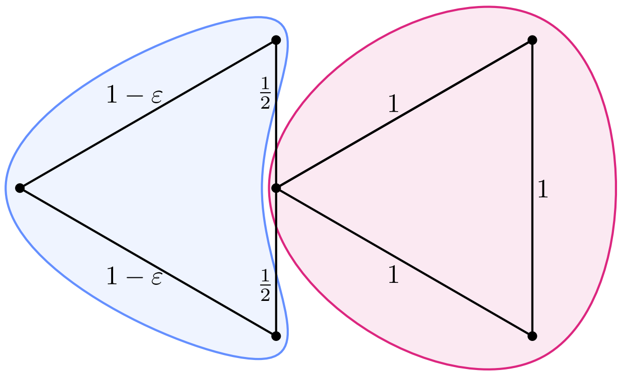


Figure 2 Visual proof of Lemma 2.4: The colored clusters are the optimal connected solution, whereas the optimal convex solution is to choose the whole graph as a cluster.

▶ **Lemma 2.5.** *Let $\alpha > 1$. There is a planar graph G such that for $r = 3$, $d^*_{\alpha-\text{ratio}} > 2d^*_{\text{conn}}$.*

Finally, we have the following relation.

▶ **Lemma 2.6.** *For each graph G with positive edge lengths and each $\alpha \geq \frac{D}{d}$, where D is the length of a longest path in G and d is the length of a shortest edge, we have $d^*_{\text{conn}} = d^*_{\alpha-\text{ratio}}$.*

Proof sketch. Let C be some cluster in the optimal connected solution and $u, v \in C$. Using that C is connected and the definition of d, D, α, we see that C is α-ratio-convex:

$$d_{G[C]}(u,v) \leq D \leq \alpha d \leq \alpha d_G(u,v). \qquad ◀$$

We conclude this section by observing that all our variants are NP-hard in general.

▶ **Lemma 2.7.** *[Corollary from [9]] The connected, α-sum-convex and α-ratio-convex and convex r-gather problems are NP-hard (for fixed $r \geq 3$).*

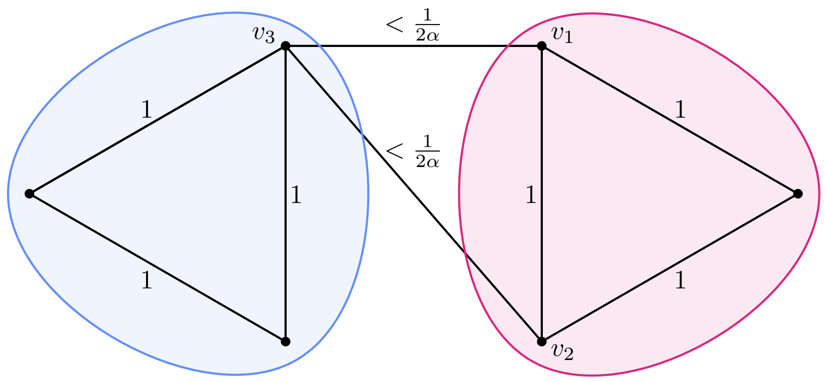


Figure 3 Visual proof of Lemma 2.5: The colored clusters are the optimal connected solution, whereas the optimal α-ratio convex solution is to choose the whole graph as a cluster

Proof sketch. The solutions to the planar circuit SAT instances constructed in [9] are convex. As the convex and unrestricted solutions are identical in this instance, all other variants are also NP-hard. ◀

3 Polynomial-Time Algorithm for Trees

We describe a dynamic program that checks for given d and $r \in \mathbb{N}$ whether a tree has a connected r-gather with each cluster (a subtree) of diameter at most d. We call such an r-gather *d-valid*. The optimal diameter d can be computed using a binary search over all possible values of d.

Let us consider a fixed diameter d and a rooted tree T. For a node u in T let T_u be the subtree rooted at u. The *depth* of a rooted tree is the number of edges of the longest path from the root to any of its leaves. For an integer $\delta \le d$, a partition of the nodes into clusters is *(δ, d, r)-valid* if the cluster containing the root (which we call the *root cluster*) is a subtree of depth at most δ and each other cluster forms a subtree of size at least r with diameter at most d. Note that a (δ, d, r)-valid partition is an r-gather if and only if the root cluster has size at least r.

In the case that there is some (δ, d, r)-valid partition of T_u, let $f_u(\delta)$ denote the largest size of the root cluster among all (δ, d, r)-valid partitions of T_u. In the case that there is no (δ, d, r)-valid partition of T_u, we distinguish two situations: if there is a d-valid r-gather of T_u (so the root cluster has size at least r as well, but no depth restriction) set $f_u(\delta) = 0$, otherwise we set $f_u(\delta) = -\infty$ (so there is no d-valid r-gather of T_u at all). In summary:

$$f_u(\delta) = \begin{cases} |C|, & \text{if a } (\delta, d, r)\text{-valid partition of } T_u \text{ exists whose root cluster} \\ & C \text{ is a largest root cluster over all } (\delta, d, r)\text{-valid partitions of } T_u, \\ 0, & \text{if there is a } d\text{-valid } r\text{-gather but no } (\delta, d, r)\text{-valid partition of } T_u, \\ -\infty, & \text{if neither a } d\text{-valid } r\text{-gather nor a } (\delta, d, r)\text{-valid partition of } T_u \text{ exists.} \end{cases}$$

The root cluster in a d-valid r-gather has depth at most d which gives the following fact.

▶ **Observation 3.1.** *A d-valid r-gather of T_u exists if and only if $f_u(d) \ge r$.*

To simplify our formulas we also define $f_u(-1) = 0$ if there is a d-valid r-gather of T_u (i.e. $f_u(d) \ge r$ by Observation 3.1) and $f_u(-1) = -\infty$ otherwise (i.e. if there is no d-valid

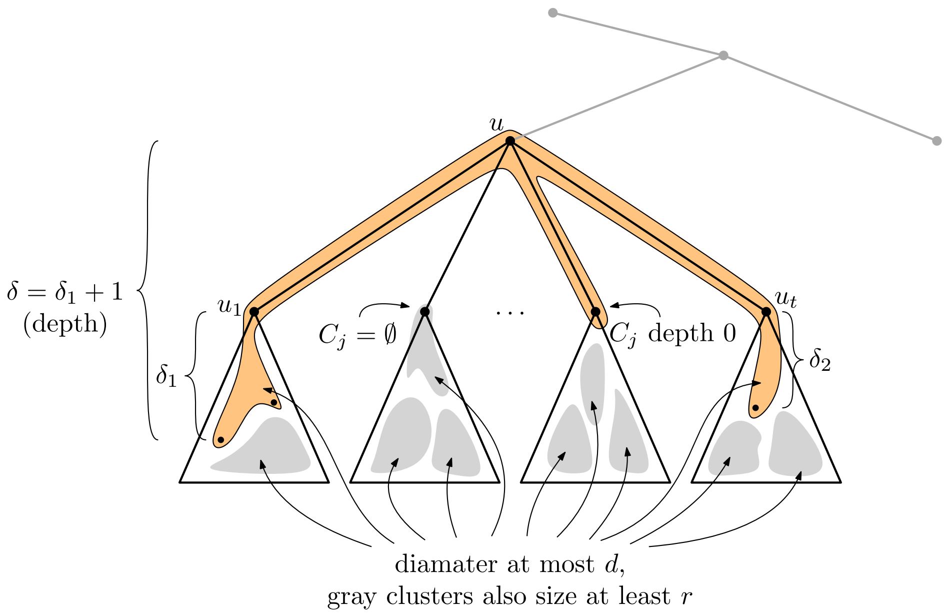


Figure 4 An illustration of a (δ, d, r)-valid partition.

r-gather of T_u at all). The value $\delta = -1$ represents the case that u shall not be part of its parent's cluster. This is also represented by the cases $f_u(\delta) = 0$.

We compute all values $f_u(\delta)$ with a dynamic program. For leaves u, $f_u(\delta) = 1$ for each $\delta \geq 0$ and $f_u(-1) = -\infty$ (as $r > 1$). Otherwise let $u_1, \dots, u_t$ be the children of u. In the recursive step, we first compute preliminary values $\widetilde{f_u}(\delta)$ for each δ with $0 \leq \delta \leq d$:

$$\widetilde{f_u}(\delta) = 1 + \max_{\substack{\delta_1,\delta_2:\\ -1\leq\delta_2\leq\delta_1\leq\delta-1,\\ \delta_1+\delta_2\leq d-2}} \max_{i=1,\dots,t} \left(f_{u_i}(\delta_1) + \sum_{j:\, j\neq i} f_{u_j}(\delta_2) \right). \tag{1}$$

We now show that $\widetilde{f_u}(\delta)$ and $f_u(\delta)$ coincide if $f_u(\delta) \neq 0$. The cases $f_u(\delta) = 0$ and $\delta = -1$ are handled in a second step below.

▶ **Lemma 3.2.** *Let u be a non-leaf node in T and $\delta \geq 0$. If $f_u(\delta) \neq 0$, then $\widetilde{f_u}(\delta) = f_u(\delta)$.*

Proof. First, consider the case that there is some (δ, d, r)-valid partition of T_u (so $f_u(\delta) > 0$). Consider a (δ, d, r)-valid partition of T_u with root cluster C of maximum size $f_u(\delta)$. For each child u_j let $C_j = C \cap V(T_{u_j})$ (possibly $C_j = \emptyset$). Let δ_1 and $\delta_2 \leq \delta_1$ denote the two largest depths among $\{C_1, \dots, C_t\}$ in the respective subtrees (it might be $\delta_1 = \delta_2$) where we use -1 for the depth of empty clusters $C_j = \emptyset$. Let C_i denote a cluster with depth δ_1. As C has depth at most δ, we have $-1 \leq \delta_2 \leq \delta_1 \leq \delta - 1$. As C has diameter at most d, we have $\delta_1 + \delta_2 \leq d - 2$ (this also works for negative δ_1, δ_2). Because C is as large as possible, each C_j is as large as possible under the depth restriction. Moreover, each C_j with $j \neq i$ is largest for depth δ_2, because δ_2 is not smaller than the depth of C_j and, hence, replacing the given partition of T_{u_j} by any (δ_2, d, r)-valid partition of T_{u_j} has root cluster of size at least $|C_j|$. Hence, $|C_i| = f_{u_i}(\delta_1)$ and $|C_j| = f_{u_j}(\delta_2)$ for each $j \neq i$. This shows

$$\begin{aligned} f_u(\delta) = |C| &= 1 + f_{u_i}(\delta_1) + \sum_{j\colon j\neq i} f_{u_j}(\delta_2) \\ &\leq 1 + \max_{\substack{\delta_1,\delta_2\colon \\ -1\leq\delta_2\leq\delta_1\leq\delta-1, \\ \delta_1+\delta_2\leq d-2}} \max_{i=1,\dots,t} \left(f_{u_i}(\delta_1) + \sum_{j\colon j\neq i} f_{u_j}(\delta_2) \right) = \widetilde{f}_u(\delta). \end{aligned}$$

Now consider integers δ_1, δ_2, and i maximizing the right hand side of the recursion (1). By the assumption of this case, there is a (δ, d, r)-valid partition of T_u. Restricting this partition to a subtree rooted at a child of T_u gives either a (δ', d, r)-valid partition of this subtree for some $0 \leq \delta' \leq \delta - 1$ or a d-valid r-gather. This implies $f_{u_i}(\delta_1) \neq -\infty$ and, for each $j \neq i$, $f_{u_j}(\delta_2) \neq -\infty$. Hence, there is a (δ_1, d, r)-valid partition of T_{u_i} with root cluster of size $f_{u_i}(\delta_1)$ or a d-valid r-gather of T_{u_i}. For each $j \neq i$ there is a (δ_2, d, r)-valid partition of T_{u_j} with root cluster of size $f_{u_j}(\delta_2)$ or a d-valid r-gather of T_{u_j}. Merging the root clusters for those children u_j with $f_{u_j}(\delta_2) > 0$ and adding u to it gives a (δ, d, r)-valid partition of T_u, as the root cluster has diameter at most $\max\{\delta_1 + \delta_2 + 2, d\} \leq d$ and depth at most $\delta_1 + 1 \leq \delta$. This shows that

$$\begin{aligned} f_u(\delta) &\geq 1 + f_{u_i}(\delta_1) + \sum_{j\colon j\neq i} f_{u_j}(\delta_2) \\ &= 1 + \max_{\substack{\delta_1,\delta_2\colon \\ -1\leq\delta_2\leq\delta_1\leq\delta-1, \\ \delta_1+\delta_2\leq d-2}} \max_{i=1,\dots,t} \left(f_{u_i}(\delta_1) + \sum_{j\colon j\neq i} f_{u_j}(\delta_2) \right) = \widetilde{f}_u(\delta). \end{aligned}$$

Now consider the case that there is neither a d-valid r-gather nor a (δ, d, r)-valid partition of T_u (so $f_u(\delta) = -\infty$). The same arguments as above show that $\widetilde{f}_u(\delta) = -\infty$ in this case. Indeed, if $\widetilde{f}_u(\delta) \neq -\infty$, then $\widetilde{f}_u(\delta) > 0$ and there are integers δ_1, δ_2 with $-1 \leq \delta_2 \leq \delta_1 \leq \delta - 1$, $\delta_1 + \delta_2 \leq d - 2$, and $f_{u_i}(\delta_1) + \sum_{j\colon j\neq i} f_{u_j}(\delta_2) \geq 0$. We then find a (δ, d, r)-valid partition of T_u by merging suitable partitions of the subtrees rooted at u's children.

This shows that $\widetilde{f}_u(\delta) = f_u(\delta)$ whenever $f_0(\delta) \neq 0$. ◀

Consider the case $f_u(\delta) = 0$, that is, there is no (δ, d, r)-valid partition of T_u but a d-valid r-gather of T_u exists. The arguments in the proof of Lemma 3.2 show that we get $\widetilde{f}_u(\delta) = -\infty$ in this case. To compensate for this, the full recursive step works as follows. First, recursion (1) is evaluated to compute preliminary values $\widetilde{f}_u(\delta)$ for each δ with $0 \leq \delta \leq d$. Then we use Observation 3.1 and distinguish two cases. If $\widetilde{f}_u(d) \geq r$, then there is a d-valid r-gather of T_u (by means of Lemma 3.2). Hence, for each $\delta \geq 0$ we set $f_u(\delta) = 0$, if $\widetilde{f}_u(\delta) = -\infty$, and $f_u(\delta) = \widetilde{f}_u(\delta)$ otherwise. We also set $f_u(-1) = 0$. If $\widetilde{f}_u(d) < r$, then there is no d-valid r-gather of T_u. So for each δ we set $f_u(\delta) = \widetilde{f}_u(\delta)$ and set $f_u(-1) = -\infty$.

Computing the preliminary values via the recursion (1) for fixed u and δ (and recursively given f values for u's children) needs time in $O(\delta^2 \deg(u))$. The update round for the values, if necessary, needs $O(d)$ time. By computing bottom-up, starting at the leaves, we compute the values of $f_u(\delta)$ for each node u in T and each $\delta = -1, \dots, d$ in time $O(d^3 n)$ (as $\delta \leq d$).

By Lemma 2.2 it suffices to search for an optimal d in the range $1, \dots, 2r-2$. This shows that the optimal d can be computed in $O(nr^3 \log r)$ time.

▶ **Theorem 3.3.** *For each unweighted tree, d^*_{conn} can be computed in $O(nr^3 \log r)$ time.*

4 Connected r-Gather Approximation

This section provides an approximation algorithm for the connected r-gather. We call it Best-or-Fill as it is very similar to the Best-or-Rest algorithm to approximate the r-gathering problem given by Armon [2]. A variation of the Best-or-Rest algorithm is presented as the distributed sweep algorithm for r-gather for a distributed setting by Zeng et al. [9]. In the "Best"-stage, for each node u we compute the smallest ball B_u (with respect to the shortest path metric) with center u containing at least r nodes. We process these clusters in sorted order by increasing diameter (breaking ties consistently but arbitrarily). We mark a cluster B_u if no node from B_u is contained in an already marked cluster and skip it otherwise. At the end of the stage, we have some marked clusters. We extend this clustering to a connected r-gather in the "Fill"-stage next. Each node u that is not contained in a marked cluster yet is assigned to a nearest marked cluster, where the distance is measured from u to a closest node in the cluster (i.e., the boundary). If several marked clusters are nearest, we take the marked cluster that was marked first. Extending the marked clusters by their respective assigned nodes gives a graph partition.

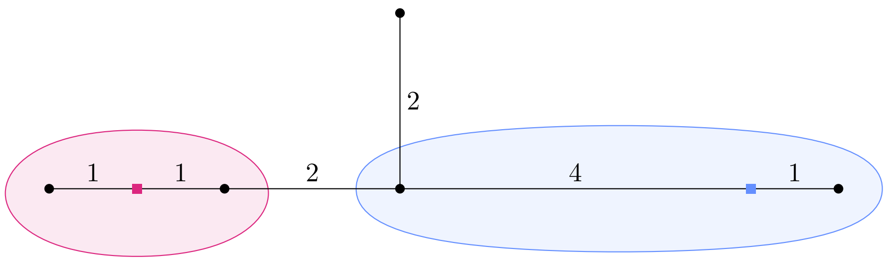


Figure 5 In Best-or-Rest, the remaining node would be assigned to the red cluster on the left.

The key difference to [2, 9] is that we need the distance to the boundary instead of the center; otherwise, the cluster may not be connected. See Figure 5.

▶ **Theorem 4.1.** *Best-or-Fill is a 4-approximation for connected r-gather on weighted graphs.*

Proof. We can construct a path through the center of the balls to prove the approximation guarantee for the diameter. The same argument with more details can be found in [9]. We only need to show connectedness, which is clear for the marked clusters from the "Best"-stage. If v gets assigned to B_u in the "Fill"-stage, then all nodes w from the shortest path from v to the boundary node x of B_u are (1) not contained in a marked cluster and (2) are assigned to B_u. (1) is true because otherwise, w would be our destination instead of x. And (2) is true because the shortest path from w to the boundary of B_u is contained in the shortest path from v to that cluster. Hence, each extended cluster is connected. ◀

References

1 Gagan Aggarwal, Rina Panigrahy, Tomás Feder, Dilys Thomas, Krishnaram Kenthapadi, Samir Khuller, and An Zhu. Achieving anonymity via clustering. *ACM Transactions on Algorithms (TALG)*, 6(3):1–19, 2010. doi:10.1145/1798596.1798602.

2 Amitai Armon. On min–max r-gatherings. *Theoretical Computer Science*, 412(7):573–582, 2011. Selected papers from WAOA 2007: Fifth Workshop on Approximation and Online Algorithms. doi:10.1016/j.tcs.2010.04.040.

3 Danilo Artigas, Simone Dantas, Mitre Costa Dourado, and Jayme Luiz Szwarcfiter. Partitioning a graph into convex sets. *Discrete Mathematics*, 311(17):1968–1977, 2011. doi:10.1016/J.DISC.2011.05.023.

4 Maike Buchin and Lukas Plätz. Anonymous routing using minimum capacity clustering (short paper). In *12th International Conference on Geographic Information Science (GIScience 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.GISCIENCE.2023.18.

5 Derya Celik Turkoglu and Müjde Erol Genevois. A comparative survey of service facility location problems. *Annals of Operations Research*, 292(1):399–468, 2020. doi:10.1007/S10479-019-03385-X.

6 Soh Kumabe and Takanori Maehara. PTAS and exact algorithms for r-gathering problems on tree. preprint, 2019. arXiv:1907.04087.

7 Soh Kumabe and Takanori Maehara. r-gathering problems on spiders: Hardness, FPT algorithms, and ptases. In *WALCOM: Algorithms and Computation: 15th International Conference and Workshops*, volume 12635 of *LNCS*, pages 154–165. Springer, 2021. doi:10.1007/978-3-030-68211-8_13.

8 Ignacio M. Pelayo. *Geodesic convexity in graphs*, volume 577 of *SpringerBriefs in Mathematics*. Springer, New York, 2013. doi:10.1007/978-1-4614-8699-2.

9 Jiemin Zeng, Gaurish Telang, Matthew P Johnson, Rik Sarkar, Jie Gao, Esther M Arkin, and Joseph SB Mitchell. Mobile r-gather: Distributed and geographic clustering for location anonymity. In *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 7:1–7:10, 2017. doi:10.1145/3084041.3084056.

Hitting Affine Families of Polyhedra, with Applications to Robust Optimization

Jean Cardinal[1], **Xavier Goaoc**[2], **and Sarah Wajsbrot**[3]

1 **Université libre de Bruxelles (ULB), Brussels, Belgium**
jean.cardinal@ulb.be

2 **Université de Lorraine, CNRS, INRIA, LORIA, Nancy, F-54000, France**
xavier.goaoc@loria.fr

3 **Université de Lorraine, CNRS, INRIA, LORIA, Nancy, F-54000, France**
sarah.wajsbrot@loria.fr

Abstract

Geometric hitting set problems, in which we seek a smallest set of points that collectively hit a given set of ranges, are ubiquitous in computational geometry. Most often, the set is discrete and is given explicitly. We propose new variants of these problems, dealing with continuous families of polyhedra, and show that they capture decision versions of the two-level finite adaptability problem in robust optimization. We show that these problems can be solved in strongly polynomial time when the size of the hitting/covering set and the dimension of the polyhedra and the parameter space are constant. This leads to new tractability results for finite adaptability that are the first ones with so-called left-hand-side uncertainty, where the underlying problem is non-linear.

1 Introduction

In this paper we investigate hitting and covering problems for *continuous* families of polyhedra and show that they capture decision versions of the two-level finite adaptability problem in robust optimization.

1.1 Hitting affine families of polyhedra

An *affinely parameterized family of polyhedra* in $\mathbb{R}^d$, or an *affine family of polyhedra* for short, is a continuous family $P(\Omega)$ of polyhedra in $\mathbb{R}^d$ defined by a domain $\Omega \subset \mathbb{R}^p$ and two affine maps $A : \Omega \mapsto \mathbb{R}^{m\times d}$ and $b : \Omega \mapsto \mathbb{R}^m$ via

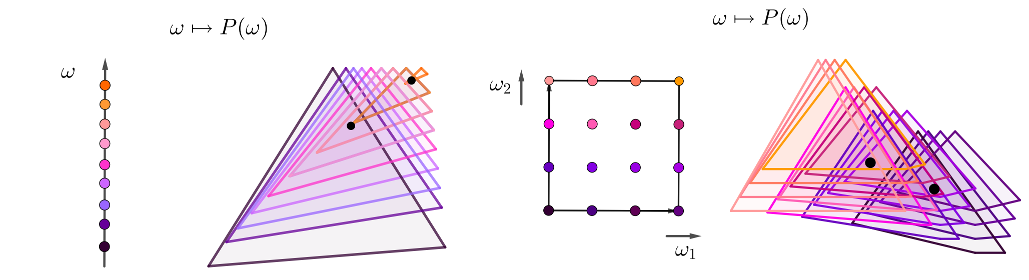
$$P(\Omega) = \{P(\omega) \colon \omega \in \Omega\} \quad \text{where} \quad P(\omega) = \left\{x \in \mathbb{R}^d \colon A(\omega)x \leq b(\omega)\right\}. \tag{1}$$

Observe that an affine family of polyhedra has three defining parameters: the dimension d of the ambient space of the polyhedra, the dimension p of the parameter space Ω, and the number m of constraints defining each polyhedron. When every polyhedron $P(\omega)$ is bounded, we call $P(\Omega)$ an *affine family of polytopes*. See Figure 1 for examples.

We call a set $S \subset \mathbb{R}^d$ a *hitting set* for an affine family of polyhedra if S intersects every member of that family. See Figure 1 for examples. We adopt the following convention: if an affine family of polyhedra $P(\Omega)$ has one empty member, that is if $P(\omega) = \emptyset$ for some $\omega \in \Omega$, then $P(\Omega)$ has no hitting set. We first consider the following computational problem:

Given $k \in \mathbb{N}$ and an affine family $\mathcal{F}$ of polyhedra, does $\mathcal{F}$ admit a hitting set of size k?

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.



$$A(\omega) = \begin{pmatrix} 0.4\omega & -1 \\ -\sqrt{3}(1-0.4\omega) & 1 \\ \sqrt{3}(1-0.4\omega) & 1 \end{pmatrix} \quad b(\omega) = \begin{pmatrix} -0.4\omega \\ 0 \\ \sqrt{3} \end{pmatrix} \qquad A(\omega) = \begin{pmatrix} 1-0.4\omega_1 & -1 \\ -\sqrt{3}(1-0.4\omega_1) & (1-0.4\omega_2) \\ \sqrt{3}(1-0.4\omega_1) & (1-0.4\omega_2) \\ 0.4\omega_2 & -1 \end{pmatrix} \quad b(\omega) = \begin{pmatrix} -0.4\omega_2 \\ -\omega_2 \\ \sqrt{3} \\ -0.4\omega_1 \end{pmatrix}$$

Figure 1 Two examples of affine families of polygons; For readability, we represent the polygon $P(\omega)$ for only a finite subset of ω in the domain.

1.2 Context: robust optimization and finite adaptability

This continuous hitting set problem arises as the decision version of a special case of the *finite adaptability* problem in robust optimization [3]. This line of research in mathematical programming deals with uncertainty in planning by modeling the decision to make as the optimization of some objective function under some constraints, with the objective function and the constraints depending not only of the free variables, but also of some uncertainty parameter ω. One then searches for an optimal solution that is valid for all values of the uncertainty parameter ω. Robust optimization problems with successive stages of decision are of particular interest [18], such as the *two-stage robust optimization problem*:

$$\begin{aligned} \inf_{x_f \in \mathbb{R}^\ell} \quad \sup_{\omega \in \Omega} \quad \inf_{x_s \in \mathbb{R}^\delta} \quad & {c_f}^T x_f + c_s(\omega)^T x_s \\ \text{s. t.} \quad & A_f(\omega) x_f + A_s(\omega) x_s \leq b(\omega) \end{aligned} \tag{2}$$

where Ω is the domain of uncertainty, $A_f(\omega)$, $A_s(\omega)$, $b(\omega)$ and $d(\omega)$ are input matrices and vectors depending on the uncertainty parameter ω, and c is a deterministic input vector. The variables x_f and x_s correspond to the first and second stages of the optimization, where the second stage takes place only once the uncertainty ω has been revealed. The problem is therefore to optimize in the first stage the worst-case outcome from the second stage.

In 2010, Bertsimas and Caramanis [4] proposed to approximate Problem (2) by the following *finite adaptability* problem:

$$\begin{aligned} \inf_{\substack{x_f \in \mathbb{R}^\ell \\ x_s^1, x_s^2, \ldots, x_s^k \in \mathbb{R}^\delta}} \quad \sup_{\omega \in \Omega} \quad \inf_{i \in [k]} \quad & {c_f}^T x_f + c_s(\omega)^T x_s^i \\ \text{s. t.} \quad & A_f(\omega) x_f + A_s(\omega) x_s^i \leq b(\omega) \end{aligned} \tag{3}$$

(We use $[k]$ to denote the set $\{1, 2, \ldots, k\}$.) For fixed k this problem is called the *k-adaptability problem*. It models the computation, in the first stage, of k candidate values for the second-stage variable x_s. Once the uncertainty ω is revealed, the second stage consists of selecting one of the k precomputed values that satisfies the constraints and minimizes the objective. Under a suitable continuity assumption, the value of Problem (3) converges to the value of (2) as $k \to \infty$ [12, § 2]. See [10, 16, 6, 17] for recent work on Problem (3) and variants.

In what follows, we assume that A_f, A_s, b, and c_s are affine maps, and consider only this version of the finite adaptability problem.

1.3 Contributions

The continuous hitting problem stated above corresponds to the decision version of the optimization Problem (3) with no first-stage decision ($\ell = 0$); See Lemma 2.1. Like in the discrete setting, the hitting and covering problems for affine families of polyhedra enjoy some form of duality. We prove (Lemma 2.2) that the decision version of the general finite adaptability problem is a special case of the following covering problem:

Given two affine families of polyhedra in $\mathbb{R}^d$ and $k \in \mathbb{N}$, does there exist k polyhedra in the second family whose union covers a polyhedron from the first family?

Note that this problem is linear, whereas Problem (3) exhibits some non-linearity. Quantifier elimination methods can solve these problems in strongly polynomial time when k and the dimensions are fixed (Theorem 3.1). Similar statements hold for the hitting and covering problems (see Section 3). A natural question is whether these problems are *fixed-parameter tractable* with respect to k and the dimensions; in the full version of this paper, we show that this is the case for 1-dimensional parameter domain and no first decision ($\ell = 0$).

1.4 Background and related works

Continuous families of geometric sets are ubiquitous in computational geometry, most notably in *range searching* [1] and the theory of *ε-nets* [11, 7, 5, 9, 8]. Geometric computation with uncertainty were also investigated by computational geometers [13]. Hitting set problems were extensively investigated in discrete and computational geometry for discrete, unstructured families of sets. One typically expects that deciding whether k points suffice to hit n given subsets of $\mathbb{R}^d$ is NP-hard when k or d is part of the input [14, 15]. When both k and d are fixed, the problem can be solved in polynomial time via the computation of an arrangement.

What about the tractability of finite adaptability? On the negative side, unless $\mathsf{P} = \mathsf{NP}$, there is no polynomial-time algorithm already in the special case where $k = 2$, there is no first decision x_f, and A_s is independent of ω [4, Prop. 5]; that proof requires the dimensions of x_s and ω as well as the number m of rows in A_s to be part of the input. On the positive side, we are only aware of two previous results: when A_f, A_s and b are constant (but c_s still depends on ω [17, Prop. B.3] and when $k \leq 3$, the number of vertices (and dimension) of Ω is bounded, and only the right-hand side b of the constraints depends on the uncertainty parameter (that is, A_f, A_s and d are constant) [12, Theorems 1.1 and 1.2]. Theorem 3.1 therefore identifies new tractable cases; note that the map A_s is affine (*left-hand-side uncertainty*), which makes the underlying problem non-linear in x_f, x_s and ω.

2 Relation between hitting/covering and finite adaptability

Let us first consider the k-adaptability problem (3) without first decision ($\ell = 0$), that is

$$\begin{aligned} \inf_{x_s^1, x_s^2, \ldots, x_s^k \in \mathbb{R}^\delta} \quad \sup_{\omega \in \Omega} \quad \inf_{i \in [k]} \quad & c_s(\omega)^T x_s^i \\ \text{s. t.} \quad & A_s(\omega) x_s^i \leq b(\omega) \end{aligned} \tag{4}$$

▶ **Lemma 2.1.** *For any real t, the value of the k-adaptability problem without first decision* (4) *is at most t if and only if the affine family of polyhedra $P_t(\Omega) = \{P_t(\omega)\colon \omega \in \Omega\}$ defined by*

$$P_t(\omega) = \left\{x \in \mathbb{R}^d \colon \begin{pmatrix} A_s(\omega) \\ c_s(\omega)^T \end{pmatrix} x \leq \begin{pmatrix} b(\omega) \\ t \end{pmatrix}\right\}$$

admits a hitting set of size k.

Proof. Let $t \in \mathbb{R}$. The value of Problem (4) is at most t if and only if there exists $x_s = (x_s^1, x_s^2, \dots, x_s^k) \in (\mathbb{R}^d)^k$ such that $\sup_{\omega\in\Omega} \inf_{i\in[k] \text{ s.t. } A_s(\omega)x_s^i \leq b(\omega)} c_s(\omega)^T x_s^i(\omega)$ is at most t. This is equivalent to the condition that for every $\omega \in \Omega$, $\inf_{i\in[k] \text{ s.t. } A_s(\omega)x_s^i \leq b(\omega)} c_s(\omega)^T x_s^i(\omega)$ is at most t. This, in turn, is equivalent to the condition that for every $\omega \in \Omega$, there exists $i \in [k]$ such that $A_s(\omega)x_s^i \leq b(\omega)$ and $c_s(\omega)^T x_s^i(\omega) \leq t$; In other words, $x_s^i \in P_t(\omega)$. Hence, the value of Problem (4) is at most t if and only if $P_t(\Omega)$ admits a hitting set of size k. Observe that in particular, Problem (4) is infeasible if and only if there is no feasible solution $(x_s^1, x_s^2, \dots, x_s^k)$, that is no hitting set of size k for $P(\Omega)$. ◀

When there is a first decision ($\ell > 0$), one may proceed as in Lemma 2.1, *mutatis mutandis*, and obtain that the value of Problem (3) is at most t if and only if there exists $x_f \in \mathbb{R}^\ell$ such that the affine family of polyhedra $P_{x_f,t}(\Omega) = \{P_{x_f,t}(\omega)\colon \omega \in \Omega\}$ defined by

$$P_{x_f,t}(\omega) = \left\{x_s \in \mathbb{R}^\delta \colon \begin{pmatrix} A_s(\omega) \\ c_s(\omega)^T \end{pmatrix} x_s \leq \begin{pmatrix} b(\omega) - A_f(\omega)x_f \\ t - {c_f}^T x_f \end{pmatrix}\right\} \tag{5}$$

admits a hitting set of size k. To simplify this formulation, we fix ℓ and p and, taking inspiration from the Veronese map, define a "lifting":

$$L : \left\{ \begin{array}{rcl} \mathbb{R}^{\ell+p} & \to & \mathbb{R}^{\ell+p+\ell p} \\ (x_f, \underbrace{\omega_1, \omega_2, \dots, \omega_p}_{\omega}) & \mapsto & (x_f, \omega, \omega_1 x_f, \omega_2 x_f, \dots, \omega_p x_f). \end{array} \right.$$

For better readability, we let $d = \ell + p + \ell p$. We use z to denote a point in $\mathbb{R}^d$, and write $z_{x_f} = (z_1, \dots, z_\ell)^T$ and $z_\omega = (z_{\ell+1}, \dots, z_{\ell+p})^T$. For $\omega \in \Omega$, we decompose $A_f(\omega)$ into $A_f(\omega) = A_{L,0} + \sum_{i\in[p]} \omega_i A_{L,i}$, where $A_{L,0}, A_{L,1}, \dots, A_{L,p} \in \mathbb{R}^{m\times\delta}$.

▶ **Lemma 2.2.** *For any real t, the value of the k-adaptability problem* (3) *is at most t if and only if there is a member of $\breve{P}_t(\mathbb{R}^\ell)$ that can be covered by some k members of $\breve{Q}_t(\mathbb{R}^\delta)$, where $\breve{P}_t(\mathbb{R}^\ell)$ is the affine family of polyhedra in $\mathbb{R}^d$ defined by $\breve{P}_t(\mathbb{R}^\ell) = \{\breve{P}_t(x_f)\colon x_f \in \mathbb{R}^\ell\}$ where $\breve{P}_t(x_f) = L(x_f \times \Omega)$, and $\breve{Q}_t(\mathbb{R}^\delta)$ is the affine family of polyhedra in $\mathbb{R}^d$ defined by $\breve{Q}_t(\mathbb{R}^\delta) = \{\breve{Q}_t(x_s)\colon x_s \in \mathbb{R}^\delta\}$ where*

$$\breve{Q}_t(x_s) = \left\{ z \in \mathbb{R}^d \colon \left\{ \begin{array}{r} A_{L,0} z_{x_f} + A_s(z_\omega)x_s + \sum_{i\in[p]} A_{L,i}(z_{ip+\ell+1}, \dots, z_{ip+\ell+p})^T \leq b(z_\omega) \\ {c_f}^T z_{x_f} + c_s(z_\omega)x_s \leq t \end{array} \right. \right\}.$$

Proof. First, note that for any fixed $t \in \mathbb{R}$, $\breve{P}_t(\mathbb{R}^\ell)$ and $\breve{Q}_t(\mathbb{R}^\delta)$ are affine families of polyhedra. For $\breve{Q}_t(\mathbb{R}^\delta)$, this is because $A_s(\omega)$ depend affinely on ω and $A_{L,0}, A_{L,1}, \dots, A_{L,p}$ are constant matrices. For $\breve{P}_t(\mathbb{R}^\ell)$, this is because Ω is a polyhedron. Next, observe that $L(\mathbb{R}^{\ell+p})$ is the surface $(\Sigma) \subseteq \mathbb{R}^d$ of dimension $\ell + p$ defined by the quadratic equations $z_{p+i\ell+j} = z_j z_{\ell+i}$ for $1 \leq i \leq p$ and $1 \leq j \leq \ell$. Letting $\pi : (z_1, z_2, \dots, z_d) \mapsto (z_1, z_2, \dots, z_{\ell+p})$ be the projection that forgets the last ℓp coordinates, we see that L induces a bijection onto (Σ) with inverse $\pi_{|(\Sigma)}$.

Let us fix $t \in \mathbb{R}$ and recall the definition of $P_{x_f,t}(\omega)$ from Equation (5). The value of problem (3) is at most t if and only if there exists $(x_f, x_s^1, x_s^2, \dots, x_s^k) \in \mathbb{R}^\ell \times (\mathbb{R}^\delta)^k$ such that for every $\omega \in \Omega$, there exists $i \in [k]$ such that $x_s^i \in P_{x_f,t}(\omega)$. Letting $S_t(x_s) = \big\{(x_f, \omega) \in \mathbb{R}^\ell \times \Omega \colon x_s \in P_{x_f,t}(\omega)\big\}$, this becomes: there exists $(x_f, x_s^1, x_s^2, \dots, x_s^k) \in \mathbb{R}^\ell \times (\mathbb{R}^\delta)^k$ such that $\{x_f\} \times \Omega$ is covered by $S_t(x_s^1) \cup S_t(x_s^2) \cup \cdots \cup S_t(x_s^k)$.

Let us now have the lift L act on this characterization. First, L is a bijection from $\mathbb{R}^{\ell+p}$ to (Σ), so $\{x_f\} \times \Omega$ is covered by $S_t(x_s^1) \cup S_t(x_s^2) \cup \cdots \cup S_t(x_s^k)$ if and only if $L(\{x_f\} \times \Omega)$ is covered by $L(S_t(x_s^1)) \cup L(S_t(x_s^2)) \cup \cdots \cup L(S_t(x_s^k))$. On the one hand, $L(\{x_f\} \times \Omega) = \breve{P}(x_f)$ by definition of $\breve{P}$. On the other hand, by definition of $\breve{Q}$ we have $L(S_t(x_s)) = \breve{Q}_t(x_s) \cap (\Sigma)$. The previous condition is therefore equivalent to the existence of $(x_f, x_s^1, x_s^2, \dots, x_s^k) \in \mathbb{R}^\ell \times (\mathbb{R}^\delta)^k$ such that $\breve{P}(x_f)$ is covered by $\Big(\breve{Q}_t(x_s^1) \cap (\Sigma)\Big) \cup \Big(\breve{Q}_t(x_s^2) \cap (\Sigma)\Big) \cup \cdots \cup \Big(\breve{Q}_t(x_s^k) \cap (\Sigma)\Big)$. Since for every $x_f \in \mathbb{R}^\ell$, the set $\breve{P}(x_f)$ is contained in (Σ), we can drop the intersections with (Σ) in that condition, and the statement follows. ◀

3 Polynomial complexity bounds

The decision version of the k-adaptability Problem (3) can be expressed using a first-order formula, whose validity can be determined by general algorithms for quantifier elimination and existential theory of the reals.

▶ **Theorem 3.1.** *For every constant k, ℓ, δ, p, there is an algorithm that solves the decision version of the k-adaptability Problem (3) on m constraints, with Ω a polyhedron in $\mathbb{R}^p$ given as an intersection of v halfspaces, in time strongly polynomial in m and v.*

Proof. The decision version of Problem 3 consists in deciding, given some real number t, whether the optimal value of (3) is at most t. This can be cast as deciding the validity of the formula $\exists x_f \in \mathbb{R}^\ell, x_s^1, x_s^2, \dots, x_s^k \in \mathbb{R}^\delta \quad \Phi(x_f, x_s)$, where

$$\Phi(x_f, x_s) \equiv \forall \omega \in \mathbb{R}^p \ (\omega \notin \Omega) \vee \big((x_f, x_s^1) \in P_t(\omega)\big) \vee \big((x_f, x_s^2) \in P_t(\omega)\big) \vee \cdots \vee \big((x_f, x_s^k) \in P_t(\omega)\big),$$

and $P_t(\Omega)$ is the affine family of polyhedra defined by

$$P_t(\omega) = \left\{(x_f, x_s) \in \mathbb{R}^{\ell+\delta} \colon \begin{pmatrix} A_f(\omega) A_s(\omega) \\ c_f(\omega)^T c_s(\omega)^T \end{pmatrix} (x_f, x_s) \le \begin{pmatrix} b(\omega) \\ t \end{pmatrix} \right\}.$$

Note that the polyhedron $P_t(\omega)$ encode both the constraints enforced by A_f, A_s, and b on (x_f, x_s^i), and the bound t on the optimal value (this is similar to the proof of Lemma 2.1). The universal quantifier on ω takes care of the $\sup_{\omega \in \Omega}$ in the formulation (3), while the disjunction on the terms of the form $(x_f, x_s^i) \in P_t(\omega)$ takes care of the $\inf_{i \in [k]}$.

We next eliminate the universal quantifier in $\Phi(x_f, x_s)$. This quantifier applies to p variables and there are $s \le v + k(m+1)$ polynomials involved, defining the polyhedra Ω and $P_t(\omega)$. Each polynomial is of degree at most $r = 2$, and the total number of variables is $k\delta + \ell$. Hence, the elimination of the quantifier can be done by a strongly polynomial algorithm of complexity $O\left((v+m)^{(k\delta+\ell+1)(p+1)}\right)$ [2, §14, Theorem 14.16]. It produces a formula with a single existential quantifier on (x_f, x_s), involving $O\left((v+m)^{(k\delta+\ell+1)(p+1)}\right)$ polynomials, each of degree is at most $2^{O(p)}$. The validity of that formula can be decided by a strongly polynomial algorithm [2, §13, Theorem 13.13] of complexity

$$\left(O\left((v+m)^{(k\delta+\ell+1)(p+1)}\right)\right)^{k\delta+\ell+1} 2^{O((k\delta+\ell)p)} = O\left((v+m)^{(k\delta+\ell+1)^2(p+1)}\right)$$

as claimed. ◀

The same method yields, for every constant k, p, and d, an algorithm that decides in time strongly polynomial in m, given a polyhedron Ω in $\mathbb{R}^p$ defined by m constraints and an affine family of polyhedra $P(\Omega)$ in $\mathbb{R}^d$, each defined by at most m constraints, whether $P(\Omega)$ has a hitting set of size k; the complexity is $m^{O(1)}$, where the exponent depends on k, p, and d. Similarly, for every constant k, γ, λ and d there is an algorithm that decides in time strongly polynomial in m, given an affine family of polyhedra $P(\Gamma)$ in $\mathbb{R}^d$ defined by at most m constraints, and an affine family of polyhedra $Q(\Lambda)$ in $\mathbb{R}^d$, defined by at most m constraints, with $\Gamma \subseteq \mathbb{R}^\gamma$ and $\Lambda \subseteq \mathbb{R}^\lambda$ polyhedra defined by at most m constraints each, whether there exist $\gamma_0 \in \Gamma$ and $\lambda_1, \lambda_2, \ldots, \lambda_k \in \Lambda$ such that $P(\gamma_0) \subseteq Q(\lambda_1) \cup Q(\lambda_2) \cup \cdots \cup Q(\lambda_k)$. The complexity is $m^{O(1)}$, where the exponent depends on k, γ, λ and d.

References

1 Pankaj K. Agarwal. Range searching. In *Handbook of Discrete and Computational Geometry*, pages 1057–1092. Chapman and Hall/CRC, 2017.

2 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer, 2006. 2nd ed. URL: https://hal.science/hal-01083587.

3 Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*, volume 28 of *Princeton Series in Applied Mathematics*. Princeton University Press, 2009. doi:10.1515/9781400831050.

4 Dimitris Bertsimas and Constantine Caramanis. Finite adaptability in multistage linear optimization. *IEEE Trans. Autom. Control.*, 55(12):2751–2766, 2010. doi:10.1109/TAC.2010.2049764.

5 Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discret. Comput. Geom.*, 14(4):463–479, 1995. doi:10.1007/BF02570718.

6 Christoph Buchheim and Jannis Kurtz. Min-max-min robust combinatorial optimization. *Math. Program.*, 163(1-2):1–23, 2017. doi:10.1007/S10107-016-1053-Z.

7 Bernard Chazelle, Herbert Edelsbrunner, Michelangelo Grigni, Leonidas J. Guibas, Micha Sharir, and Emo Welzl. Improved bounds on weak epsilon-nets for convex sets. *Discret. Comput. Geom.*, 13:1–15, 1995. doi:10.1007/BF02574025.

8 Khaled Elbassioni. A bicriteria approximation algorithm for the minimum hitting set problem in measurable range spaces. *Oper. Res. Lett.*, 51(5):507–514, 2023. doi:10.1016/J.ORL.2023.07.005.

9 Guy Even, Dror Rawitz, and Shimon Shahar. Hitting sets when the VC-dimension is small. *Inf. Process. Lett.*, 95(2):358–362, 2005. doi:10.1016/J.IPL.2005.03.010.

10 Grani Adiwena Hanasusanto, Daniel Kuhn, and Wolfram Wiesemann. *K*-adaptability in two-stage robust binary programming. *Oper. Res.*, 63(4):877–891, 2015. doi:10.1287/OPRE.2015.1392.

11 David Haussler and Emo Welzl. Epsilon-nets and simplex range queries. *Discret. Comput. Geom.*, 2:127–151, 1987. doi:10.1007/BF02187876.

12 Safia Kedad-Sidhoum, Anton Medvedev, and Frédéric Meunier. Finite adaptability in two-stage robust optimization: asymptotic optimality and tractability, 2024. arXiv:2305.05399.

13 Maarten Löffler. *Data Imprecision in Computational Geometry.* PhD thesis, Utrecht University, Netherlands, 2009.

14 Nimrod Megiddo. On the complexity of some geometric problems in unbounded dimension. *J. Symb. Comput.*, 10(3/4):327–334, 1990. doi:10.1016/S0747-7171(08)80067-3.

15 Nimrod Megiddo and Kenneth J. Supowit. On the complexity of some common geometric location problems. *SIAM J. Comput.*, 13(1):182–196, 1984. doi:10.1137/0213014.

16 Krzysztof Postek and Dick den Hertog. Multistage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. *INFORMS J. Comput.*, 28(3):553–574, 2016. doi:10.1287/IJOC.2016.0696.

17 Anirudh Subramanyam, Chrysanthos E. Gounaris, and Wolfram Wiesemann. K-adaptability in two-stage mixed-integer robust optimization. *Math. Program. Comput.*, 12(2):193–224, 2020. *Extended version:* arXiv:1706.07097. doi:10.1007/S12532-019-00174-2.

18 Ihsan Yanikoglu, Bram L. Gorissen, and Dick den Hertog. A survey of adjustable robust optimization. *Eur. J. Oper. Res.*, 277(3):799–813, 2019. doi:10.1016/J.EJOR.2018.08.031.

On Triangular Separation of Bichromatic Point Sets

Helena Bergold[1], Arun Kumar Das[2], Robert Lauff[3], Manfred Scheucher[3], Felix Schröder[4], and Marie Diana Sieper[5]

1 Technische Universität München
helena.bergold@gmail.com
2 Czech Technical University
arund426@gmail.com
3 Technische Universität Berlin
lauff@math.tu-berlin.de,scheucher@math.tu-berlin.de
4 Charles University
schroder@kam.mff.cuni.cz
5 JMU Würzburg
marie.sieper@uni-wuerzburg.de

Abstract

We address the problem of computing the minimum number of triangles to separate a set of blue points from a set of red points in $\mathbb{R}^2$. A set of triangles is a *separator* of one color from the other if every point of that color is contained in some triangle and no triangle contains points of both colors. We consider several variants of the problem depending on whether the triangles are allowed to overlap or not and whether all points or just the blue points need to be contained in a triangle. We show that computing the minimum cardinality triangular separator of a set of blue points from a set of red points is NP-hard and further investigate worst case bounds on the minimum cardinality of triangular separators for a bichromatic set of n points.

1 Introduction

Separability problems on colored point sets are concerned with computing a class of objects of given shapes for a given set of colored points, such that every point of a fixed color is covered with the computed objects and no object contains points of different colors. The exact problem varies depending on the underlying optimizing criteria. We study triangles separating red and blue points in $\mathbb{R}^2$. Separating red and blue points from each other is well studied in computational geometry under the name *class cover problems* [2, 5, 8] due to their wide applications in data mining, pattern recognition, learning theory, and operations research. The problem is formally defined as follows:

▶ Problem 1 (Overlapping-Separation). **Input:** A set of red and blue points $P \subset \mathbb{R}^2$
Output: A minimum set $\mathcal{T}$of triangles covering all points such that no member of $\mathcal{T}$contains points of both colors

Note that this problem can be subdivided into two problems. First find the minimum number of triangles $\mathcal{T}_b$ covering all the blue points in P such that no triangle in $\mathcal{T}_b$ contains a red point. Then compute the minimum number of triangles $\mathcal{T}_r$ covering the red, but no blue points. The set $\mathcal{T}_b \cup \mathcal{T}_r$ is an optimal solution for Problem 1. Thus it reduces to:

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

▶ Problem 2 (Overlapping-Separation-of-blue). **Input:** A set of red and blue points $P \subset \mathbb{R}^2$ **Output:** A minimum set of triangles covering all blue but no red points of P

However once the triangles are not allowed to intersect each other, the problem becomes more restricted and spawns two different problems for separating the points as follows.

▶ Problem 3 (Disjoint-Separation). **Input:** A set of red and blue points $P \subset \mathbb{R}^2$ **Output:** A minimum set $\mathcal{T}$of disjoint triangles covering all points such that no member of $\mathcal{T}$contains points of both colors

▶ Problem 4 (Disjoint-Separation-of-blue). **Input:** A set of red and blue points $P \subset \mathbb{R}^2$ **Output:** A minimum set of disjoint triangles covering all blue but no red points of P

Researchers addressed the problem of computing the minimum number of separators for a bichromatic point set. Canon and Cowen [5] first considered the problem of finding the minimum number of circles to separate a bichromatic point set in a general metric space. They proved that computing the minimum number of circles centered at the blue points to separate them from the red ones is NP-hard and presented a $(\ln n + 1)$-factor approximation algorithm and devised a PTAS for the problem in $\mathbb{R}^d$. Bereg et al. [3] showed that computing the minimum number of axis-parallel rectangles to separate a bichromatic planar point set is NP-hard. In the same paper, they addressed the problem of separating objects by vertical or horizontal strips and presented a $\mathcal{O}(r \log r + b \log b + \sqrt{rb})$-time exact algorithm where r and b are the numbers of red and blue points respectively. They also proved separation to be NP-hard if the separating objects are half-strips/squares and presented $\mathcal{O}(1)$-approximations. But separation and partition problems are not only interesting on the algorithmic side. Motivated by a question of Aharoni and Saks, Dumitrescu et al. [9, 10, 11] showed that every bichromatic set of n points can be partitioned into $\lfloor \frac{n}{2} \rfloor + 1$ monochromatic subsets with disjoint convex hulls. This is not true if the subsets have a maximum size of 2. They give an algorithm to find a matching of size $\frac{3}{7}n$, but show that a monochromatic matching can in some cases only cover $\frac{94}{95}n$ points. This is in contrast to the classic problem of Putnam that every bichromatic set of n blue and n red points admits a perfect matching of the red and the blue points ([12], for a proof of some generalization, see for instance [1]). The special case of a maximum size of a matching stabbed by a line with points on a circle has been of both long-term and recent interest under the name ”necklace folding problem”[6, 13].

	cover both	cover blue
disjoint	$\geq \lfloor \frac{n}{2} \rfloor + 1$ (Proposition 1)	$\geq \lfloor \frac{n}{4} \rfloor + 1$ (Proposition 2)
	$\leq \lfloor \frac{n}{2} \rfloor + 1$ (Proposition 3)	$\leq \lfloor \frac{2}{7}n \rfloor + 1$ (Proposition 5)
overlap	$\geq \frac{3}{8}n - \mathcal{O}(1)$ (Proposition 2)	$\geq \lfloor \frac{n}{4} \rfloor + 1$ (Proposition 2)
	$\leq \frac{13}{30}n + \mathcal{O}(1)$ (Proposition 6)	$\leq \frac{4}{15}n + \mathcal{O}(1)$ (Proposition 4)

Table 1 Bounds on triangles separating bichromatic planar point sets of n points in total.

We show that Overlapping-Separation-of-blue and Disjoint-Separation-of-blue are NP-hard. Then we prove combinatorial bounds on the number of triangles required for the problems in the worst case for n points, see Table 1. The paper is organized in the following manner. Section 2 describes the hardness of the problems by a polynomial time reduction from the Planar Monotone 3-SAT problem. Section 3 presents an overview of the combinatorial results. Finally, the paper is concluded in Section 4. For detailed proofs we refer to the full version [4].

2 NP-hardness

In this section we present a polynomial-time reduction from the known NP-hard Planar Monotone (PM) 3-SAT [7] problem to the problems Overlapping-Separation-of-blue and Disjoint-Separation-of-blue (Problems 2 and 4). PM 3-SAT is a special version of 3-SAT, where the usual boolean formula ϕ is in conjunctive normal form, each clause contains at most 3 literals which are either all positive or all negative. Moreover, there exists a planar embedding Γ_ϕ of the incidence graph of ϕ, in which the variables lie on the x-axis, the clauses with positive literals are above the x-axis and the clauses with negative literals are below the x-axis. Given a PM 3-SAT formula ϕ, we construct a planar bichromatic point set such that we can decide whether ϕ is satisfiable based on the number of triangles in a minimum separator. The condition holds whether the triangles are allowed to overlap or not. Thus we simultaneously address the NP-hardness of problems 2 and 4.

▶ **Theorem 2.1.** Overlapping-Separation-of-blue *and* Disjoint-Separation-of-blue *are NP-hard.*

Proof sketch. Here we give a simplified reduction for brevity. For the full formal details, see the full version [4]. Given a PM 3-SAT formula ϕ with k variables $x_1, \dots, x_k$, m_2 clauses with 2 literals and m_3 clauses with 3 literals as well as a corresponding planar embedding Γ_ϕ, we will construct a bichromatic point set such that all blue points can be covered with $t := k + m_2 + 2m_3$ triangles not containing any red points, if and only if ϕ is satisfiable. If the cover is possible, it will be possible inside these *covering* triangles:

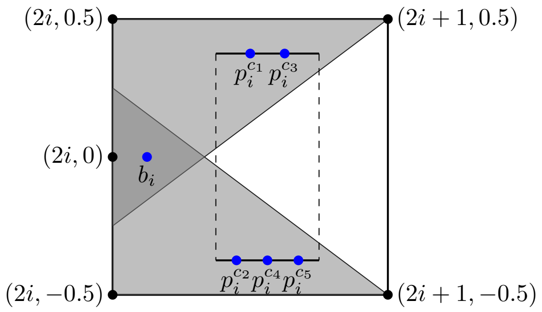


Figure 1 The covering triangles of the variable gadget of x_i are marked in grey. The white space between the dashed lines can be used for red points.

Let $\varepsilon > 0$ be small enough. First we describe how to replace the variable-vertices of Γ_ϕ with variable gadgets. The variable x_i is represented by vertices in a square. Inside it, place two covering triangles intersecting in a small triangle next to the point $(2i, 0)$ and place a blue point b_i in it. For every clause c involving x_i, place a point p_i^c on a top or bottom segment close enough to the middle as in Figure 1 making sure that they are ordered according to the order of incidences of the clauses at x_i in Γ_ϕ. Use the top segment if and only if c is positive. Replace the vertex associated to the positive clause $c = x_i \vee x_j \vee x_k, i < j < k$ of Γ_ϕ with 2 blue points l_c and r_c, shifted slightly to the left and right, respectively. Place four covering triangles covering each of the pairs of points $(l_c, p_i^c), (l_c, p_j^c), (r_c, p_j^c)$ and (r_c, p_k^c) so closely that no other point is inside and triangles only intersect if their pairs do. If $c = x_i \vee x_j, i < j$ has only two literals, its point $l_c = r_c$ is considered blue and we introduce two covering triangles covering it and one of p_i^c or p_j^c. Negative clauses are handled analogously. Then perturb the placed blue points to establish general position.

Two of the previously placed blue points are *incompatible* if no covering triangle contains both. For every pair of incompatible points, we place a red point on the segment between them, but not into any covering triangle. Figure 1 shows how this is done if both points are in the same variable gadget. If they are not, the triangles outside of the variable gadgets are thin enough not to contain the segment completely. In an ε-ball around each blue point, we place $t + 1$ extra blue points in such a way that general position of the blue point set is preserved. We call these points form the *blue point cloud* around the blue point. The ε-ball of every red point r is contained in the convex hull of the ε-balls of its defining blue points, but disconnects it. For every triangle of one point from one of the associated blue point clouds and two of the other, we place a red point inside it and the ε-ball of r and then delete r. Since ε is small, they are not contained in any covering triangles. We place these points such that the full point set is in general position. They constitute the *red point cloud* of r.

We now show that the thus defined point set can be covered with t triangles if and only if ϕ is satisfiable. Assume that ϕ is satisfiable. For every false variable, we use the bottom covering triangle in the variable gadget. For every true variable, we use the top one. For every positive clause $c = x_i \vee x_j \vee x_k$, we use two triangles:

1. If x_i is false, we use the triangle of (l_c, p_i^c). If x_k is false, we use the triangle of (r_c, p_k^c).
2. If both x_i and x_k are false, x_j is true, so p_j^c has been covered and we are done.
3. Else if x_j is false we use (l_c, p_j^c) or (r_c, p_j^c), depending on which of the first two triangles was not used, so l_c and r_c are not covered twice.
4. If we did not cover both l_c and r_c this way, we cover their point clouds individually.

If c has only two literals use the covering triangle containing the false literal or cover it individually. Negative clauses are handled analogously. This covers all blue points with t triangles. We only use triangles inside covering triangles, so no triangle contains a red point.

Assume now t triangles cover the blue points. Every point cloud contains more than t points, so some triangle contains at least 2 points of it. This triangle is said to *cover* the cloud. No two incompatible clouds are covered by the same triangle, otherwise it would contain a red point. However the points l_c, r_c for all clauses c and the points b_i for all variables x_i are pairwise incompatible, so we need all t triangles for them, k *variable-covering* and $m_2 + 2m_3$ *clause-covering*. If the triangle covering b_i covers any p_i^c for a positive clause c, set x_i to true, else to false. Since any clause c has one less triangle covering it than literals, $\exists x_i \in c : p_i^c$ is covered by the variable-covering triangle of x_i. Hence c is fulfilled. ◀

3 Bounds on the number of triangles

This section roughly describes our techniques for the bounds on the number of triangles needed to separate a bichromatic point set in general position in the worst case.

Lower bounds: We use two different point sets: When covering both colors with disjoint triangles, we use points along a closed convex curve, colored alternatingly with blue and red. This makes sure a triangle can only ever cover three points, however if disjoint, at least as many triangles as those that reach that bound cover only one point as well:

▶ **Proposition 1.** *Let $k \geq 0$. Given a bichromatic point set on a closed convex curve in the plane such that $2k$ is the number of times two consecutive points on the curve have different* *colors, there is no set of k disjoint triangles containing all points such that no triangle contains points of both colors.*

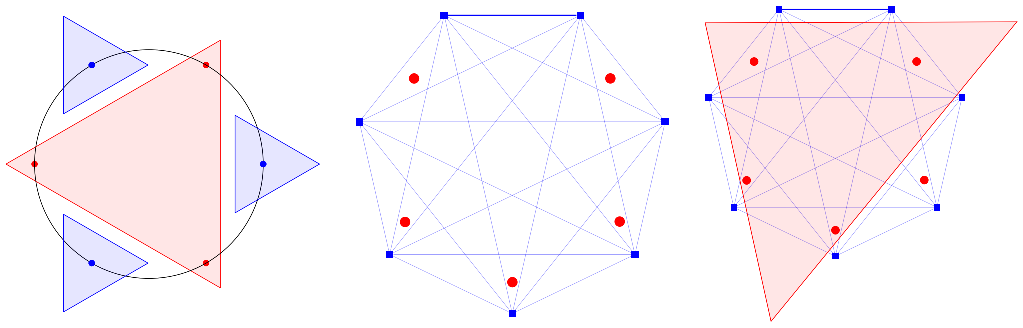


Figure 2 Illustration of lower bounds. Left: Proposition 1; Middle: Proposition 2; Right: A red triangle covering 4 red points: One of the edges cuts off ≥ 3 blue points, so this is the maximum.

In the overlap case, this would give a bound of $\lceil \frac{n}{3} \rceil$, but we can do a little bit better by considering a slightly different construction: The blue points are equidistributed along a circle and an almost equal number of red points are placed inside so that any triangle of three blue points contains one of them. This construction gives the $\frac{n}{4} + \Omega(1)$ lower bounds right away, whether the triangles are overlapping or not. Some further investigation reveals that the red points can be chosen such that no five of them can be in a triangle without blue points. This gives the final result of $\frac{1}{2}\frac{n}{2} + \frac{1}{4}\frac{n}{2} + \Omega(1) = \frac{3}{8}n + \Omega(1)$ triangles for overlapping triangles covering both point sets.

▶ **Proposition 2.** *There are bichromatic planar n-point sets such that there is no set of $\lfloor \frac{n}{4} \rfloor$ triangles containing all blue points and no set of $\frac{3}{8}n + \mathcal{O}(1)$ triangles containing all points such that no triangle contains points of both colors.*

Upper bounds: We use sweeping lines and case distinction at the boundary of the convex hull. First we prove that our lower bound for disjoint covering of all points is tight.

▶ **Proposition 3.** *Given a bichromatic planar n-point set, there are $\lfloor \frac{n}{2} \rfloor + 1$ disjoint triangles within the convex hull of P containing all points such that none contains points of both colors.*

We also give bounds for the number of triangles needed to cover all blue points in terms of the number of blue points only, then using the number of red points only to deduce:

▶ **Proposition 4.** *Given a bichromatic planar n-point set, there is a set of $\frac{4}{15}n + \mathcal{O}(1)$ vertices, segments and triangles containing no red points and using all blue points as vertices once.*

▶ **Proposition 5.** *Given a bichromatic planar n-point set, there are $\frac{2}{7}n + 1$ disjoint triangles containing all blue points such that no triangle contains a red point.*

Finally by computer, we show that the maximum number of points in a bichromatic planar point set without two vertex-disjoint empty monochromatic triangles is 14 and obtain:

▶ **Proposition 6.** *Given a bichromatic planar n-point set, there are $\frac{13}{30}n + \mathcal{O}(1)$ triangles containing all points such that no triangle contains points of both colors.*

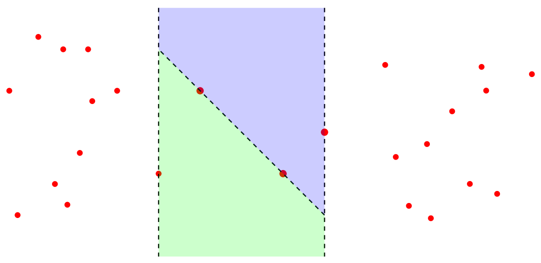


Figure 3 Sweepline construction of triangles to cover the blue points using the red points only.

4 Conclusion

We have addressed the hardness and bounds on the minimum number of triangular separators for a bichromatic point set. Theorem 2.1 proves the NP-hardness of DISJOINT-SEPARATION-OF-BLUE but it is questionable whether this problem belongs to NP at all. It might be $\exists\mathbb{R}$-hard [15].

▶ **Conjecture 1.** DISJOINT-SEPARATION-OF-BLUE *is* $\exists\mathbb{R}$*-complete.*

Table 2 contains our results how many triangles we need to use to cover $n \leq 12$ points in the four different settings. Based on this, the OEIS [14] suggests the following formulae

n	2	3	4	5	6	7	8	9	10	11	12
one color, overlap	1	1	2	2	2	2	3	3	3	3	4
one color, disjoint	1	1	2	2	2	2	3	3	3	3	4
both colors, overlap	2	2	3	3	4	4	4	5	5	6	6
both colors, disjoint	2	2	3	3	4	4	5	5	6	6	7

Table 2 Results on covering/separating small bichromatic point sets

for the overlap setting, one of which coincides with our previously proved lower bound:

▶ **Conjecture 2.** *Every planar bichromatic set of* n *points in general position can be covered by at most* $\lfloor \frac{2}{5}(n+4) \rfloor$ *monochromatic triangles.*

In the following conjecture the blue triangles could even possibly be chosen to be disjoint.

▶ **Conjecture 3.** *The blue points in every planar bichromatic set of* n *points in general position can be covered by at most* $\lfloor \frac{n}{4} \rfloor + 1$ *blue triangles.*

The main roadblock to proving this conjecture is that in a set of predominantly blue points, some triangles need to cover at least 4 blue points. However a triangle that covers 4 blue points in convex position cannot be chosen inside the convex hull of those points. Therefore a similar approach as in the proof of the $\lfloor \frac{n}{2} \rfloor + 1$ bound seems out of reach.

References

1 J. Akiyama and N. Alon. Disjoint simplices and geometric hypergraphs. *Annals of the New York Academy of Sciences*, 555(1):1–3, May 1989.

2 Bogdan Andrei Armaselu. *On the geometric separability of bichromatic point sets*. PhD thesis, UTD Theses and Dissertations, 2017.

3 Sergey Bereg, Sergio Cabello, José Miguel Díaz-Báñez, Pablo Pérez-Lantero, Carlos Seara, and Inmaculada Ventura. The class cover problem with boxes. *Computational Geometry*, 45(7):294–304, 2012.

4 Helena Bergold, Arun Kumar Das, Robert Lauff, Manfred Scheucher, Felix Schröder, and Marie Diana Sieper. On triangular separation of bichromatic point sets, 2025. URL: `https://arxiv.org/abs/2503.05178`, `arXiv:2503.05178`.

5 Adam H Cannon and Lenore J Cowen. Approximation algorithms for the class cover problem. *Annals of Mathematics and Artificial Intelligence*, 40:215–223, 2004.

6 Endre Csóka, Zoltán L. Blázsik, Zoltán Király, and Dániel Lenger. Upper bounds for the necklace folding problems. *Journal of Combinatorial Theory, Series B*, 157:123–143, 2022.

7 Mark De Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *International Journal of Computational Geometry & Applications*, 22(03):187–205, 2012.

8 Jason Gary DeVinney. *The class cover problem and its applications in pattern recognition.* PhD thesis, The Johns Hopkins University, 2003.

9 Adrian Dumitrescu and Rick Kaye. Matching colored points in the plane: Some new results. *Computational Geometry*, 19(1):69–85, 2001.

10 Adrian Dumitrescu and János Pach. Partitioning colored point sets into monochromatic parts. In *Algorithms and Data Structures*, pages 264–275. Springer Berlin Heidelberg, 2001.

11 Adrian Dumitrescu and William Steiger. On a matching problem in the plane. *Discrete Mathematics*, 211(1):183–195, 2000.

12 G. Larson. *Problem-solving Through Problems.* Springer-Verlag, 1983.

13 Rune B. Lyngsø and Christian N. S. Pedersen. Protein folding in the 2D HP model. *BRICS Report Series*, RS-99(16), 1999.

14 OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences. Published electronically at http://oeis.org.

15 Marcus Schaefer and Daniel Štefankovič. Fixed points, nash equilibria, and the existential theory of the reals. *Theory of Computing Systems*, 60(2):172–193, 2017.

Better Late than Never: the Complexity of Arrangements of Polyhedra

Boris Aronov[1], Sang Won Bae[2], Otfried Cheong[3], David Eppstein[4], Christian Knauer[5], and Raimund Seidel[6]

1 Department of Computer Science and Engineering, Tandon School of Engineering, New York University, Brooklyn, NY, USA. boris.aronov@nyu.edu
2 Division of Computer Science and Engineering, Kyonggi University, Suwon, Korea. swbae@kgu.ac.kr
3 SCALGO, Aarhus, Denmark. otfried@scalgo.com
4 Computer Science Department, University of California, Irvine, USA. eppstein@uci.edu
5 Department of Computer Science, University of Bayreuth, Germany. christian.knauer@uni-bayreuth.de
6 Saarland University, Saarland Informatics Campus, Saarbrücken, Germany. rseidel@cs.uni-saarland.de

Abstract

Let S be the subdivision of $\mathbb{R}^d$ induced by m convex polytopes having n facets in total. Assuming that the set of hyperplanes supporting these facets is in general position, we prove that S has complexity $O(m^{\lceil d/2 \rceil} n^{\lfloor d/2 \rfloor})$ and that this bound is tight. The bound is mentioned several times in the literature, but no proof for arbitrary dimension has been published before.

1 Introduction

We consider a collection of m convex polyhedra in $\mathbb{R}^d$, each given as the intersection of halfspaces, where the total number of halfspaces is $n \geqslant m$. The family of polyhedra induces a subdivision S of $\mathbb{R}^d$ into cells and faces of dimensions 0 to d. What is the complexity of this subdivision S, that is, what is the number of its vertices?

When $m = 1$, S is a single convex polyhedron defined by n halfspaces, so by the Upper Bound Theorem, its complexity is $O(n^{\lfloor d/2 \rfloor})$. At the other extreme, when $m = n$, each polyhedron is a halfspace, and S is the arrangement of n hyperplanes, which has complexity $\Theta(n^d)$.

We generalize both bounds by showing

▶ **Theorem 1.1.** *The subdivision in $\mathbb{R}^d$ induced by m convex polyhedra with a total of n facets where the set of hyperplanes supporting these facets is in general position, has complexity $O(m^{\lceil d/2 \rceil} n^{\lfloor d/2 \rfloor})$, and this bound is tight.*

This bound has been mentioned several times in the literature [2, 3, 4, 5, 6], referring to a manuscript or preprint by Aronov, Bern, and Eppstein from either 1991 or 1995. The original manuscript no longer exists, and the authors do not recollect their proof. The second edition [10] of the *Handbook of Discrete and Computational Geometry* describes the bound in Section 24.6. In the third edition, however, this section has been silently removed [11]. The chapters on arrangements by Agarwal and Sharir [1, page 58] and Pach and Sharir [8, page 19] also state the bound.

Hirata et al. [7] claim a slightly weaker result in their Lemma 4.2, but the last line of their proof does not hold in dimension larger than four.

Our proof follows the same lines as the proof of an asymptotic version of the Upper Bound Theorem by Seidel [9].

2 The upper bound

We assume that the hyperplanes defining the m polyhedra are in general position, so their arrangement is simple. We pick a generic vertical direction; in particular, no two vertices of the arrangement will be at equal altitude.

Let v be a vertex of the subdivision S. It is defined by d hyperplanes H. Let I be the set of polyhedra that contribute to H, and let U be a neighborhood of v that is small enough such that only the hyperplanes of H intersect it.

The hyperplanes H cut U into 2^d cells. One of these cells lies in the intersection P of the polytopes in I. This polytope P has d edges incident to v. As in Seidel [9], we observe that at least half of these edges either go up or down with respect to our vertical direction. Let us assume there are $i \geqslant d/2$ edges going up. Then there is a unique i-face f in P that contains those edges [9].

The i-face f lies in an i-flat F that is the intersection of $d-i$ of the hyperplanes in H. Let H' be this subset of hyperplanes. The intersection of F with P is exactly the i-face f, and v is the lowest vertex of f.

This implies that the vertex v is uniquely defined by the choice of the $d-i$ hyperplanes H' and the intersection polytope P. The polytope P, on the other hand, is uniquely defined by the at most d polytopes in I. That is, we can uniquely specify v by selecting $d-i$ hyperplanes and at most i polytopes (that do not yet contribute to those hyperplanes).

For a given $i \geqslant d/2$, there are therefore $O(n^{d-i}m^i)$ such vertices, for a total of

$$\sum_{i=\lceil d/2 \rceil}^{d} O(n^{d-i}m^i) = O(m^{\lceil d/2 \rceil} n^{\lfloor d/2 \rfloor}).$$

3 The lower bound

Recall the product construction for convex polytopes, as for instance described in Ziegler's book [12, page 10]. For polytopes $P \subset \mathbb{R}^p$ and $Q \subset \mathbb{R}^q$ the product polytope is defined to be the set

$$P \times Q = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} : x \in P,\ y \in Q \right\}.$$

This product polytope has dimension $\dim(P) + \dim(Q)$ and its nonempty faces are the products of nonempty faces of P and nonempty faces of Q. The inequalities describing the facets of $P \times Q$ are the union of the inequalities describing the facets of P (which have coefficients 0 for the "y-coordinates") and the inequalities for the facets of Q (which have coefficient 0 for the "x-coordinates"). The coordinates of the vertices of $P \times Q$ are all concatenations of the ("x") coordinates of the vertices of P with ("y") coordinates of vertices of Q. This implies that the number of facets of $P \times Q$ is the sum of the facet numbers of P and Q, whereas the number of vertices is the product of the vertex numbers.

It is easy to see that something analogous holds for facet and vertex numbers of product subdivisions:

▶ **Lemma 3.1.** *Let $P_1, \dots, P_m$ be p-dimensional polytopes with N facets in total and with V vertices in the induced subdivision. Similarly let $Q_1, \dots, Q_m$ be q-dimensional polytopes with M facets in total and with W vertices in the induced subdivision.*

The set $P_1 \times Q_1, P_2 \times Q_2, \ldots, P_m \times Q_m$ of $(p+q)$-dimensional polytopes has $N+M$ facets in total and their induced subdivision has $V \cdot W$ vertices.

Let $s > 1$ be an integer constant. For integer $\ell \geqslant 3$ let C be a regular ℓ-sided convex polygon in $\mathbb{R}^2$ with edges tangent to the unit circle. Consider the s-fold product polytope $C^s = \underbrace{C \times C \times \cdots \times C}_{s \text{ times}}$. It has $s \cdot \ell$ facets and ℓ^s vertices. For $n = s\ell$ and $d = 2s$ this is a particularly simple construction of a d-polytope with n facets and an asymptotically maximal $O(n^{\lfloor d/2 \rfloor})$ vertices.

For integer $m \geqslant 1$ and $0 \leqslant i < m$ let C_i be the polygon C rotated by $i\frac{2\pi}{\ell m}$ around the origin and let P_i be the d-polytope C_i^s, where we continue to consider even $d = 2s$.

We claim that the polytopes $P_0, \ldots, P_{m-1}$ have $sm\ell$ facets in total, and their subdivision has $(\ell \cdot m^2)^s$ vertices.

It suffices to show that the the polygons $C_0, \ldots, C_{m-1}$ have in total $m\ell$ facets and their induced subdivision has $\ell \cdot m^2$ vertices, and then repeatedly apply Lemma 3.1. The total facet number for the m polygons is clearly $m\ell$. For the vertex count in the subdivision observe that each of the $\binom{m}{2}$ pairs of the ℓ-gons have their boundaries intersect in 2ℓ points, which, including the ℓm corners yields overall $\ell \cdot m^2$ vertices.

If you let $\ell = \frac{n}{s \cdot m}$, then $P_0, \ldots, P_{m-1}$ have n facets overall, and the subdivision has

$$(\ell \cdot m^2)^s = \frac{n^s m^s}{s^s} = \Theta(n^{\lfloor d/2 \rfloor} m^{\lceil d/2 \rceil})$$

vertices if s is considered a constant and we are considering even dimension $d = 2s$.

For odd $d = 2s+1$, take the above construction, choose m intervals, say, $J_i = [-1-i, 1+i]$, choose $\ell = \frac{n-2m}{ms}$ for the construction of the P_i's above, and consider the products $Q_i = P_i \times J_i$ for $0 \leqslant i < m$. The J_i have $2m$ "facets" in total and their subdivision has $2m$ vertices. Applying Lemma 3.1 then yields that the Q_i's have n facets in total and the number of vertices in their induced subdivision is

$$2m \cdot \Theta(n^s m^s) = \Theta(n^s m^{s+1}) = \Theta(n^{\lfloor d/2 \rfloor} m^{\lceil d/2 \rceil}) .$$

4 Open problems

We needed to assume that the n hyperplanes are in general position. Does the upper bound remain true without these restriction?

One commonly uses a perturbation to prove claims on objects in degenerate position, but it is not clear how to apply this here: the difficulty is that when we perturb the hyperplanes, a feature defined by the intersection of some hyperplanes may no longer be a feature of the intersection of polytopes.

If we only want to count the *vertices* of the arrangement S, then our proof still seems to work: consider a vertex v. There may be many hyperplanes containing v, so pick a subset H of d hyperplanes whose intersection is exactly v. We then proceed as in Section 2, picking $d - i$ hyperplanes defining an i-flat F and additional i polytopes. What is now different is that the intersection polytope P of the polytopes does not necessarily intersect F in an i-flat—the intersection might be lower-dimensional, but at the very least it includes the vertex v, and that vertex is the bottom point of the intersection, so it is again uniquely specified.

Going further, the argument also works if we count vertices with multiplicity. Consider a vertex v lying on a set of hyperplanes belonging to k different polytopes. Consider all subsets of these k polytopes, and pick a subset if v is a vertex of the subarrangement of the selected polytopes, but not of any subfamily. If we count v with multiplicity equal to the number of such subsets, then our upper bound still holds.

What remains is to bound the number of j-faces of the arrangement, for $1 \leqslant j \leqslant d$, and that's where a new idea seems to be needed.

Acknowledgments

Work by Boris Aronov was partially supported by NSF Grant CCF-20-08551; part of the research was done while visiting University of Bayreuth in the summer of 2024. S.W. Bae was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT) (No. RS-2023-00251168). Research of David Eppstein was supported in part by NSF grant CCF-2212129. Research of Otfried Cheong funded in part by the European Union (ERC, KARST, project number 101071836). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

1 Pankaj K. Agarwal and Micha Sharir. Arrangements and their applications. In *Handbook of computational geometry*, pages 49–119. Elsevier, 2000.

2 Boris Aronov and Micha Sharir. The common exterior of convex polygons in the plane. *Computational Geometry*, 8:139–149, 1997.

3 Boris Aronov, Micha Sharir, and Boaz Tagansky. The union of convex polyhedra in three dimensions. *SIAM Journal on Computing*, 26:1670–1688, 1997.

4 Mark de Berg, Dan Halperin, Mark Overmars, and Marc van Kreveld. Sparse arrangements and the number of views of polyhedral scenes. *International Journal of Computational Geometry & Applications*, 7:175–195, 1997.

5 Leonidas J. Guibas, Dan Halperin, Hirohisa Hirukawa, Jean-Claude Latombe, and Randall H. Wilson. Polyhedral assembly partitioning using maximally covered cells in arrangements of convex polytopes. *International Journal of Computational Geometry & Applications*, 8:179–199, 1998.

6 Leonidas J. Guibas, Rajeev Motwani, and Prabhakar Raghavan. The robot localization problem. *SIAM Journal on Computing*, 26:1120–1138, 1997.

7 Tomio Hirata, Jiří Matoušek, Xue-Hou Tan, and Takeshi Tokuyama. Complexity of projected images of convex subdivisions. *Computational Geometry*, 4:293–308, 1994. doi:10.1016/0925-7721(94)00009-3.

8 János Pach and Micha Sharir. *Combinatorial geometry and its algorithmic applications: The Alcalá lectures*. American Mathematical Soc., 2009.

9 Raimund Seidel. The upper bound theorem for polytopes: an easy proof of its asymptotic version. *Computational Geometry*, 5:115–116, 1995. doi:10.1016/0925-7721(95)00013-Y.

10 C.D. Toth, J. O'Rourke, and J.E. Goodman. *Handbook of Discrete and Computational Geometry, Second Edition*. Discrete Mathematics and Its Applications. CRC Press, 2004. URL: https://books.google.de/books?id=X1gBshCclnsC.

11 C.D. Toth, J. O'Rourke, and J.E. Goodman. *Handbook of Discrete and Computational Geometry, Third Edition*. Discrete Mathematics and Its Applications. CRC Press, 2018. URL: https://books.google.de/books?id=X1gBshCclnsC.

12 Günter M. Ziegler. *Lectures on Polytopes*. Springer New York, NY, 2012.

Minimum-length coordinated motions for two convex centrally-symmetric robots

David Kirkpatrick[1] **and Paul Liu**[2]

1 University of British Columbia
`kirk@cs.ubc.ca`

2 Stanford University
`paul.liu@stanford.edu`

Abstract

We study the problem of determining coordinated motions, of minimum total length, for two arbitrary convex centrally-symmetric (CCS) robots in an otherwise obstacle-free plane, using the total path length traced by the two robot centres as a measure of distance. We give an exact description of shortest (but not necessarily unique) collision-avoiding motions for all initial and goal configurations of the robots.

1 Introduction

1.1 Problem Statement and Related Work

Given a collection of robots in the plane, each with specified initial and goal positions, we are interested in the problem of characterizing efficient collision-free coordinated motions taking all of the robots from their initial to their goal position. This problem has a rich history; see the full version of this paper [6] for a partial overview. More comprehensive overviews of results on the topic are provided by the recent papers of Abrahamsen and Halperin [1] and Fekete et al. [4].

In this paper, we focus on understanding shortest coordinated translational motions (hereafter called co-motions) of two convex centrally-symmetric (CCS) robots in an otherwise obstacle-free plane, using the the total path length traced by the two robot centres as a measure of distance. We note that the specification of optimal or near-optimal co-motions for two disk robots was recently identified as one of ten fundamental problems in geobotics by Abrahamsen and Halperin [1]. Our comprehensive description of shortest co-motions, for arbitrary initial and goal placements, builds on a similar characterization for the special case of disc robots developed by Kirkpatrick and Liu [7, 8]. Essentially the same techniques were used by Esteban et al. [3] to address the special case of congruent (i.e. similarly aligned) square robots. The analysis underlying the results of the current paper, as well as these earlier works, rests heavily on an application of the Cauchy surface area formula, which was introduced earlier still by Icking et al. [5], in describing optimal motions for rods (directed line segments) in the plane, where distance is measured by the length sum of the paths traced by the two endpoints of the rod. (Rod motion can be viewed as the co-motion of two discs constrained to remain in contact throughout the motion.)

Our results subsume and simplify these earlier results that addressed special cases of optimal collision-free motion planning for two robots. In doing so, they highlight both the generality and limitations of the approach initiated in [7, 8]. A critical observation is that optimal co-motions can be described in terms of the Minkowski sum of the two robots (cf. Figures 1 and 2), rather than appealing to specific geometric properties of discs or axis-aligned squares.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

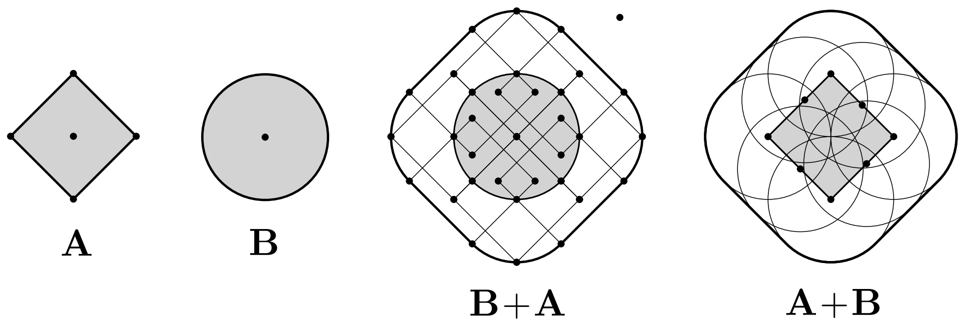


Figure 1 Two CCS objects and their Minkowski sums.

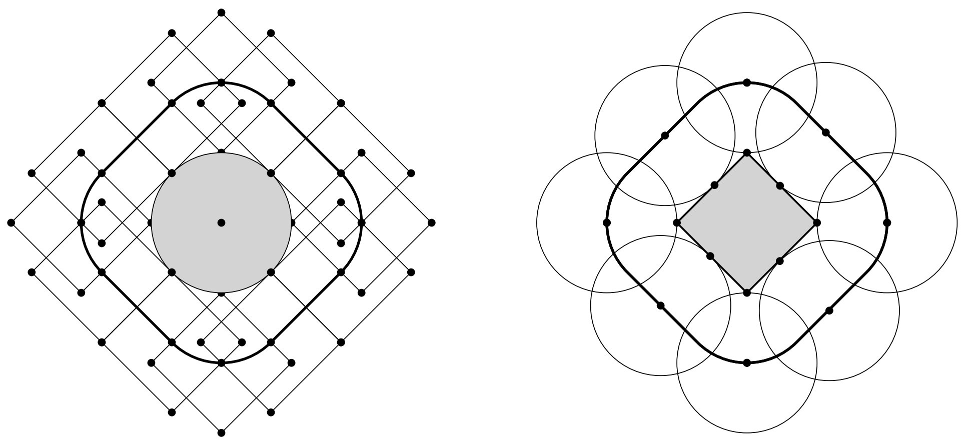


Figure 2 Minkowski sums as the locus of intersecting placements of CCS objects.

1.2 Our contributions

The coordinated motion of a pair of robots $\mathbb{A}$ and $\mathbb{B}$, from initial to goal configuration, involves either a net clockwise (abbreviated cw) or net counter-clockwise (abbreviated ccw) turn in their relative orientation. Our results describe either (i) a single globally optimal co-motion, or (ii) two co-motions (whose length could be arbitrarily similar), one of which one is optimal among all net-cw co-motions and the other is optimal among all net-ccw co-motions. For all of the optimal co-motions that we describe, the trace of both robots follows a simple standard form. In this form, the individual motions are *decoupled*, but simple modifications facilitate other, potentially desirable, properties like orientation monotonicity or contact preservation.

The rest of this extended abstract is organized as follows. We start by setting out some basic definitions as well as some fundamental tools that are used in developing our co-motions and in proving their optimality. We then summarize the general structure of our proofs, providing concrete insight by illustrating optimal motions for a representative case. Hopefully this will suffice to allow the reader to gain a solid intuitive understanding of our approach; full details are provided in [6]. Other related results mentioned above, including configuration monotonicity and contact preservation, are left to [6]. in their entirety.

1.3 Basic Definitions and Tools

We begin with the definition of a few non-standard notions; other more-intuitive notions are defined in [6]..

▶ **Definition 1.1.** The *reach* of a set of points $\mathbb{S}$ in direction θ, is given by

$$r_{\mathbb{S}}(\theta) := \sup\{x\cos\theta + y\sin\theta : (x, y) \in \mathbb{S}\}.$$

For an angle θ, the set of points that realize the supremum are called *support points*, and the line oriented at angle $\frac{\pi}{2} + \theta$ passing through the support points is called the *support line* (see Figure 3). When the set $\mathbb{S}$ consists of a single point S, we write r_S instead of $r_{\mathbb{S}}$ for ease of notation.

▶ **Observation 1.2.** *Note that $r_{\mathbb{S}}(\pi + \theta) = r_{-\mathbb{S}}(\theta)$. Furthermore, $r_{\mathbb{S}}(\theta) = r_{\widehat{\mathbb{S}}}(\theta)$, where $\widehat{\mathbb{S}}$ denotes the boundary of the convex hull of $\mathbb{S}$.*

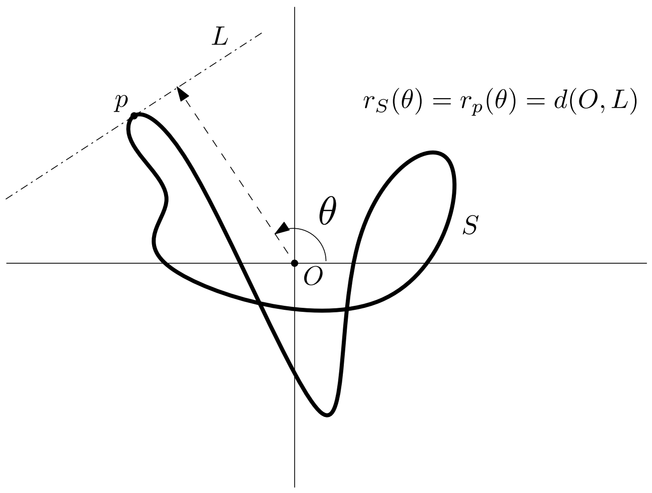


Figure 3 Support line L and support point p defining the reach of S in direction θ.

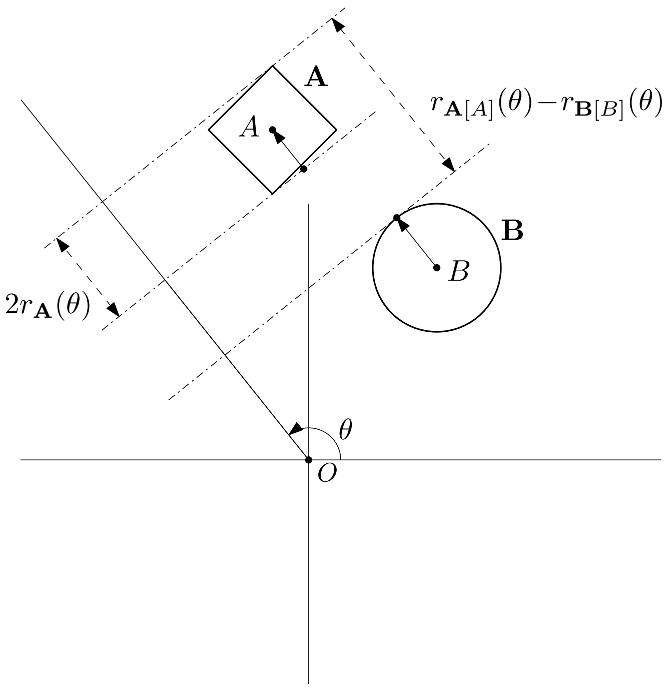
.

▶ **Definition 1.3.** A generic *CCS robot*, hereafter simply *robot*, is an open convex and centrally-symmetric region in the plane. The *placement* of robot $\mathbb{Z}$ at position Z in $\Re^2$, denoted $\mathbb{Z}[Z]$, is formed by translating $\mathbb{Z}$ so that its centre coincides with Z. Accordingly, $r_{\mathbb{Z}[Z]}(\theta) = r_Z(\theta) + r_{\mathbb{Z}}(\theta)$.

▶ **Definition 1.4.** A configuration $(\mathbb{A}[A], \mathbb{B}[B])$ is *viable* if $\mathbb{A}[A] \cap \mathbb{B}[B] = \emptyset$; equivalently if the *separation* of $\mathbb{A}[A]$ from $\mathbb{B}[B]$ in direction θ (see Figure 4) is non-negative for some direction θ. We refer to the direction θ that maximizes the separation of $\mathbb{A}[A]$ from $\mathbb{B}[B]$ as its *orientation.*

A co-motion of a robot pair takes an initial configuration to a goal configuration, through viable intermediate configurations:

▶ **Definition 1.5.** A (translational) *motion* $\xi_{\mathbb{Z}}$ of a robot $\mathbb{Z}$ from placement $\mathbb{Z}[Z_0]$ (i.e. position Z_0) to placement $\mathbb{Z}[Z_1]$ (i.e. position Z_1) is a continuous, rectifiable curve of the form $\xi_{\mathbb{Z}} : [0, 1] \to \Re^2$, where $\xi_{\mathbb{Z}}(0) = Z_0$, and $\xi_{\mathbb{Z}}(1) = Z_1$. The set of points $tr(\xi_{\mathbb{Z}}) = \{\xi_{\mathbb{Z}}(t) \mid 0 \le t \le 1\}$ is called the *trace* of robot $\mathbb{Z}$ under motion $\xi_{\mathbb{Z}}$. The *length* of motion $\xi_{\mathbb{Z}}$, denoted $\ell(\xi_{\mathbb{Z}})$, is simply the Euclidean arc-length of its trace.



A
A
$r_{\mathbf{A}[A]}(\theta) - r_{\mathbf{B}[B]}(\theta)$
B
B
$2r_{\mathbf{A}}(\theta)$
θ
O

Figure 4 The separation of $\mathbb{A}[A]$ from $\mathbb{B}[B]$ in direction θ is $r_{\mathbb{A}[A]}(\theta) - r_{\mathbb{B}[B]}(\theta) - 2r_{\mathbb{A}}(\theta)$.

▶ **Definition 1.6.** A *coordinated motion m* of a robot pair $(\mathbb{A}, \mathbb{B})$ from an initial configuration $(\mathbb{A}[A_0], \mathbb{B}[B_0])$ to a goal configuration $(\mathbb{A}[A_1], \mathbb{B}[B_1])$ is a pair $(\xi_{\mathbb{A}}, \xi_{\mathbb{B}})$, where $\xi_{\mathbb{A}}$ (resp. $\xi_{\mathbb{B}}$) is a motion of $\mathbb{A}$ (resp. $\mathbb{B}$) from placement $\mathbb{A}[A_0]$ to placement $\mathbb{A}[A_1]$ (resp. placement $\mathbb{B}[B_0]$ to placement $\mathbb{B}[B_1]$). The co-motion m is said to be *collision-free* if the configuration $(\mathbb{A}[\xi_{\mathbb{A}}(t)], \mathbb{B}[\xi_{\mathbb{B}}(t)])$ is viable, for all $t \in [0,1]$. Co-motion $m = (\xi_{\mathbb{A}}, \xi_{\mathbb{B}})$ is said to be *convex* if both $\xi_{\mathbb{A}}$ and $\xi_{\mathbb{B}}$ are convex curves. Its length, denoted $\ell(m)$, is the sum of the lengths of its associated motions, i.e. $\ell(m) = \ell(\xi_{\mathbb{A}}) + \ell(\xi_{\mathbb{B}})$.

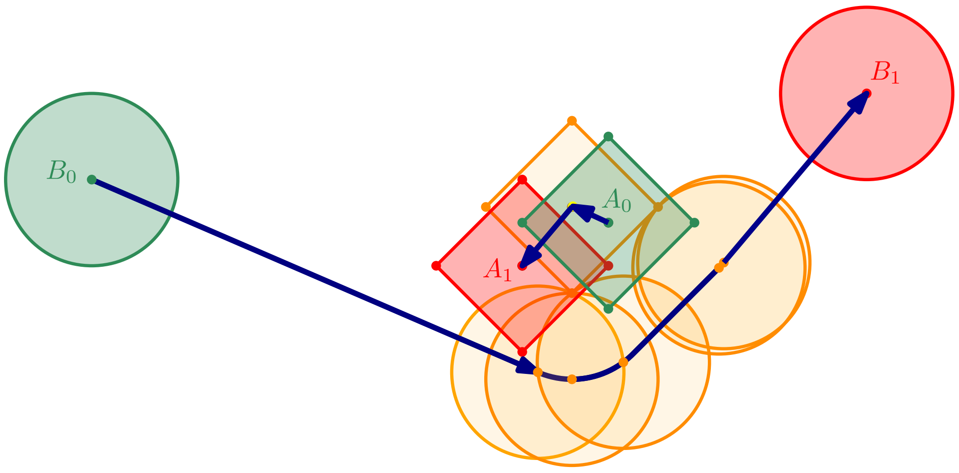


Figure 5 The trace of a co-motion of the robot pair $(\mathbb{A}, \mathbb{B})$ from initial configuration $(\mathbb{A}[A_0], \mathbb{B}[B_0])$ (green) to target configuration $(\mathbb{A}[A_1], \mathbb{B}[B_1])$ (red), with intermediate configurations (yellow)

.

Figure 5 illustrates a co-motion (blue) of two robots from their initial configuration (green) to a goal configuration (red). The diamond robot first moves (straight) to its intermediate placement (orange). Then the disc robot moves around the diamond robot to its goal placement, passing through intermediate placements with boundary contact. Finally, the diamond robot moves (straight) to its goal placement.

2 The general approach

2.1 Net counter-clockwise co-motions

We consider co-motions $(\xi_{\mathbb{A}}, \xi_{\mathbb{B}})$ that take a specified CCS robot pair $(\mathbb{A}, \mathbb{B})$ from a specified initial configuration $(\mathbb{A}[A_0], \mathbb{B}[B_0])$ (denoted $\mathbb{AB}_0$), with orientation θ_0, to a specified goal configuration $(\mathbb{A}[A_1], \mathbb{B}[B_1])$ (denoted $\mathbb{AB}_1$), with orientation θ_1. Note that, by the continuity of $\xi_{\mathbb{A}}$ and $\xi_{\mathbb{B}}$, the range of configuration orientations realized by any co-motion between $\mathbb{AB}_0$ and $\mathbb{AB}_1$ is either net-ccw (containing $[\theta_0, \theta_1]$) or net-cw (containing $S^1 - (\theta_0, \theta_1)$), or both. It suffices to restrict attention to minimum length net-ccw co-motions , since net-cw co-motions become net-ccw co-motions when reflected, together with the initial and goal configurations, across an arbitrary line.

2.2 Determining motion length

For any convex motion of $\mathbb{Z}$ from Z_0 to Z_1, $\ell(\xi_{\mathbb{Z}}) = \ell(\widehat{\xi_{\mathbb{Z}}}) - |\overline{Z_0 Z_1}|$, where $\widehat{\xi_{\mathbb{Z}}}$ denotes the boundary of the convex hull of $\xi_{\mathbb{Z}}$. Thus, to determine the length of a convex co-motion $(\xi_{\mathbb{A}}, \xi_{\mathbb{B}})$ it suffices to determine the length of the pair of closed convex curves $(\widehat{\xi_{\mathbb{A}}}, \widehat{\xi_{\mathbb{B}}})$. For this we follow an approach—a direct application of Cauchy's surface area formula [2, Section 5.3]), first described by Icking et al. [5]—that allows us to express the the length of two convex curves in terms of the reach of those curves in all directions.

▶ **Lemma 2.1.** $\ell(\widehat{\xi_{\mathbb{A}}}) + \ell(\widehat{\xi_{\mathbb{B}}}) = \int_{S^1} (r_{\widehat{\xi_{\mathbb{A}}}}(\theta) + r_{\widehat{\xi_{\mathbb{B}}}}(\pi + \theta)) d\theta.$

Note that $r_{\widehat{\xi_{\mathbb{A}}}}(\theta) + r_{\widehat{\xi_{\mathbb{B}}}}(\pi + \theta)$ describes the maximum separation, in direction θ, realized by a point on $\widehat{\xi_{\mathbb{A}}}$ and a point on $\widehat{\xi_{\mathbb{B}}}$. The following observations allow us to give a lower bound (realizable, as it turns out) on the length of optimal co-motions:

▶ **Observation 2.2.** *If θ lies in the range of orientations realized by the set of all configurations associated with co-motion m, then for some $t \in [0,1]$, the configuration $(\mathbb{A}[\xi_{\mathbb{A}}(t)], \mathbb{B}[\xi_{\mathbb{B}}(t)])$ has orientation θ, and hence by the viability property and central-symmetry, $r_{\widehat{\xi_{\mathbb{A}}}}(\theta) + r_{\widehat{\xi_{\mathbb{B}}}}(\pi + \theta) \geq r_{\xi_{\mathbb{A}}(t)}(\theta) + r_{\xi_{\mathbb{B}}(t)}(\pi + \theta) \geq r_{\mathbb{A}}(\theta) + r_{\mathbb{B}}(\pi + \theta) = r_{\mathbb{A} - \mathbb{B}}(\theta) = r_{\mathbb{A}+\mathbb{B}}(\theta)$.*

▶ **Observation 2.3.** *Since the reach of $\widehat{\xi_{\mathbb{A}}}$ is at least the reach of the segment $\overline{A_0 A_1}$ and the reach of $\widehat{\xi_{\mathbb{B}}}$ is at least the reach of the segment $\overline{B_0 B_1}$ it follows that $r_{\widehat{\xi_{\mathbb{A}}}}(\theta) + r_{\widehat{\xi_{\mathbb{B}}}}(\pi + \theta) \geq r_{\overline{A_0 A_1}}(\theta) + r_{\overline{B_0 B_1}}(\pi + \theta)$.*

The pair $(\widehat{\xi_{\mathbb{A}}}, \widehat{\xi_{\mathbb{B}}})$ is said to be *ccw-tight (with respect to the pair* $(\mathbb{AB}_0, \mathbb{AB}_1)$*)* if for all $\theta \in S^1$,

$$r_{\widehat{\xi_{\mathbb{A}}}}(\theta) + r_{\widehat{\xi_{\mathbb{B}}}}(\pi + \theta) = \max\left(r_{\overline{A_0 A_1}}(\theta) + r_{\overline{B_0 B_1}}(\pi + \theta), r_{\mathbb{A}+\mathbb{B}}(\theta) \cdot \mathbb{1}_{[\theta_0, \theta_1]}(\theta) \right), \tag{1}$$

where $\mathbb{1}_{[\theta_0, \theta_1]}$ is the indicator function of the range $[\theta_0, \theta_1]$. By the Observations above,

$$\ell(m) \geq \int_{S^1} \max\left(r_{\overline{A_0 A_1}}(\theta) + r_{\overline{B_0 B_1}}(\pi + \theta), r_{\mathbb{A}+\mathbb{B}}(\theta) \cdot \mathbb{1}_{[\theta_0, \theta_1]}(\theta) \right) \mathrm{d}\theta - |\overline{A_0 A_1}| - |\overline{B_0 B_1}|, \tag{2}$$

Thus, by Lemma 2.1, it follows that the ccw-tightness of $(\widehat{\xi_{\mathbb{A}}}, \widehat{\xi_{\mathbb{B}}})$ is a sufficient condition for any convex net-ccw co-motion $(\xi_{\mathbb{A}}, \xi_{\mathbb{B}})$ to be ccw-optimal.

▶ **Observation 2.4.** *To demonstrate a ccw-optimal co-motion* $(\xi_{\mathbb{A}}, \xi_{\mathbb{B}})$ *taking* $\mathbb{A}\mathbb{B}_0$ *to* $\mathbb{A}\mathbb{B}_1$ *it suffices to prove (i) the convexity of both* $\xi_{\mathbb{A}}$ *and* $\xi_{\mathbb{A}}$ *(typically trivial) and (ii) the ccw-tightness of* $(\widehat{\xi_{\mathbb{A}}}, \widehat{\xi_{\mathbb{B}}})$*. For the latter, we consistently show that* $r_{\widehat{\xi_{\mathbb{A}}}}(\theta)$ *is determined by either* A_0 *or* A_1*, and* $r_{\widehat{\xi_{\mathbb{B}}}}(\pi+\theta)$ *is determined by either* B_0 *or* B_1 *(and hence* $r_{\widehat{\xi_{\mathbb{A}}}}(\theta) + r_{\widehat{\xi_{\mathbb{B}}}}(\pi+\theta) = r_{\overline{A_0A_1}}(\theta) + r_{\overline{B_0B_1}}(\pi+\theta)$*), except for angles* θ *(all of which lie in the range* $[\theta_0, \theta_1]$*), where* $r_{\widehat{\xi_{\mathbb{A}}}}(\theta) + r_{\widehat{\xi_{\mathbb{B}}}}(\pi+\theta) = r_{\mathbb{A}+\mathbb{B}}(\theta)$*.*

2.3 Standard-form co-motions

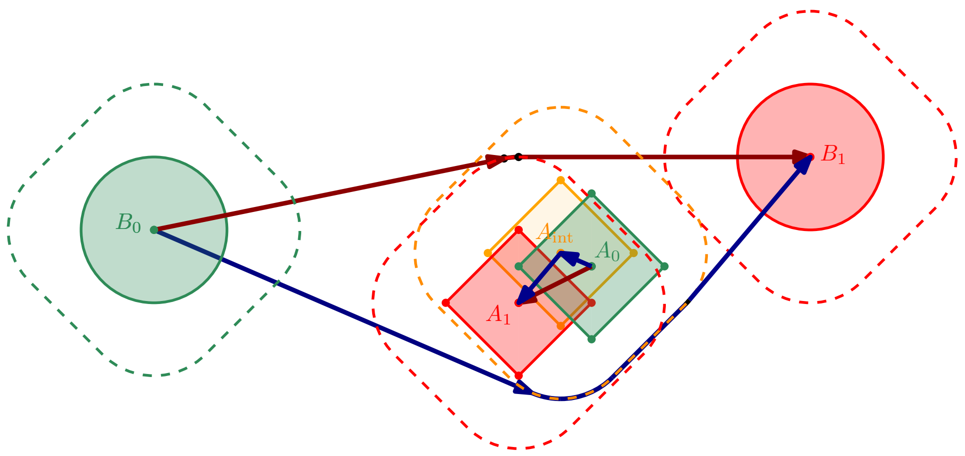


Figure 6 The trace of two collision-free co-motions from the configuration $(\mathbb{A}[A_0], \mathbb{B}[B_0])$ (green) to the configuration $(\mathbb{A}[A_1], \mathbb{B}[B_1])$ (red). The dark blue co-motion is net counter-clockwise and the dark red co-motion is net clockwise.

Figure 6 illustrates two co-motions from the configuration $(\mathbb{A}[A_0], \mathbb{B}[B_0])$ (green) to the configuration $(\mathbb{A}[A_1], \mathbb{B}[B_1])$ (red). The blue co-motion consists of three sub-motions: (i) first $\mathbb{A}$ translates from placement $\mathbb{A}[A_0]$ to placement $\mathbb{A}[A_{\text{int}}]$ (yellow); (ii) next $\mathbb{B}$ translates from placement $\mathbb{B}[B_0]$ to placement $\mathbb{B}[B_1]$, avoiding $\mathbb{A}[A_{\text{int}}]$; (iii) finally $\mathbb{A}$ translates from placement $\mathbb{A}[A_{\text{int}}]$ to placement $\mathbb{A}[A_1]$. The red co-motion consists of just two sub-motions: (i) first $\mathbb{A}$ translates from placement $\mathbb{A}[A_0]$ to placement $\mathbb{A}[A_1]$; (ii) next $\mathbb{B}$ moves from placement $\mathbb{B}[B_0]$ to placement $\mathbb{B}[B_1]$, avoiding placement $\mathbb{A}[A_1]$. As it happens, the blue co-motion is ccw-optimal and the red co-motion is cw-optimal.

The blue co-motion illustrated in Figure 6 (as well as the red co-motion, viewed from the opposite side of the host plane) is a special case of what we call *standard-form* net-ccw co-motions:

1. Move $\mathbb{A}$ from position A_0 to some intermediate position A_{int}, along a ccw-oriented shortest path that avoids $\mathbb{B}[B_0]$;
2. Move $\mathbb{B}$ from position B_0 to position B_1, along a ccw-oriented shortest path that avoids $\mathbb{A}[A_{\text{int}}]$;
3. Move $\mathbb{A}$ from position A_{int} to position A_1, along a ccw-oriented shortest path that avoids $\mathbb{B}[B_1]$.

It turns out that in looking for globally optimal net-ccw co-motions we can restrict attention to those in standard-form. The proof involves (i) the identification of a suitable choice for the point A_{int} in all possible cases, and (ii) the demonstration that for the resulting co-motion $m = (\xi_{\mathbb{A}}, \xi_{\mathbb{B}})$ the pair $(\widehat{\xi_{\mathbb{A}}}, \widehat{\xi_{\mathbb{B}}})$ is ccw-tight. See [6], section 6, for the details of this case analysis. In those cases where m is convex, it must be ccw-optimal. In those cases where m is not convex we show that any globally optimal co-motion must be cw-optimal.

2.4 Establishing counter-clockwise tightness

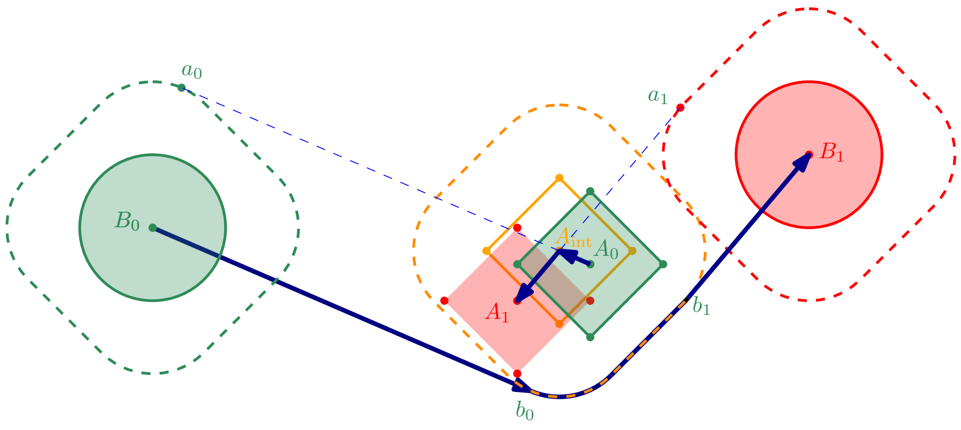


Figure 7 The trace of an optimal ccw co-motion from the configuration $\mathbb{A}\mathbb{B}_0$ (green) to the configuration $\mathbb{A}\mathbb{B}_1$ (red).

It is instructive to examine a special case, illustrated in Figure 7, one of the co-motions illustrated previously in Figure 6. This case provides one of the simplest non-trivial examples of a ccw-optimal (though not globally optimal, as it happens) co-motion, as well as an illustration of the form of the proofs that support our claims of optimality.

Let a_0 and a_1 be the upper tangent points from A_0 to $(\mathbb{A}+\mathbb{B})[B_0]$, and A_1 to $(\mathbb{A}+\mathbb{B})[B_1]$ respectively. These two tangents intersect in a point A_{int}. Let b_0 and b_1 be the lower tangent points from B_0 and B_1 to $(\mathbb{A}+\mathbb{B})[A_{\text{int}}]$ respectively. Note that by construction, the tangent from A_0 to a_0 passes through A_{int} and by symmetry is parallel to the tangent from B_0 to b_0 (Similarly, the tangent from A_1 to a_1 is parallel to the tangent from B_1 to b_1.)

▶ **Observation 2.5.** *The proof that the pair $(\widehat{\xi_{\mathbb{A}}}, \widehat{\xi_{\mathbb{B}}})$ associated with the co-motion $m = (\xi_{\mathbb{A}}, \xi_{\mathbb{B}})$ in the example above is ccw-tight, follows a pattern, supported by Observation 2.4, that we will use throughout in our consideration of convex and net-ccw co-motions m, taking initial configuration $\mathbb{A}\mathbb{B}_0$ to goal configuration $\mathbb{A}\mathbb{B}_1$. Specifically, observe that, $r_{\widehat{\xi_{\mathbb{A}}}}(\theta)$ is determined by either A_0 or A_1, and $r_{\widehat{\xi_{\mathbb{B}}}}(\pi+\theta)$ is determined by either B_0 or B_1 (and hence $r_{\widehat{\xi_{\mathbb{A}}}}(\theta)+r_{\widehat{\xi_{\mathbb{B}}}}(\pi+\theta) = r_{\overline{A_0A_1}}(\theta)+r_{\overline{B_0B_1}}(\pi+\theta)$) except for angles θ (all of which lie in the range $[\theta_0, \theta_1]$), where $r_{\widehat{\xi_{\mathbb{A}}}}(\theta) + r_{\widehat{\xi_{\mathbb{B}}}}(\pi+\theta) = r_{\mathbb{A}+\mathbb{B}}(\theta)$*

3 Conclusion

In those cases where the ccw-tight standard-form co-motion that we identify is convex, the co-motion is ccw-optimal, by Observation 2.4. In those cases where the identified co-motion is not convex we show (cf. Section 4.2) in [6]. that any globally optimal co-motion must be cw-optimal.

Taken together this provides a constructive proof of our main result:

▶ **Theorem 3.1.** *For any given initial and goal configurations, there is a globally optimal co-motion $m^* = (\xi^*_{\mathbb{A}}, \xi^*_{\mathbb{B}})$ of standard form. The length of m^*, and hence any globally optimal motion, satisfies $\ell(m^*) = \ell(\widehat{tr(\xi^*_{\mathbb{A}}) - tr(\xi^*_{\mathbb{B}})}) - |\overline{A_0A_1}| - |\overline{B_0B_1}|$.*

Acknowledgments. We acknowledge, with appreciation, several helpful discussions with Dan Halperin.

References

1 Mikkel Abrahamsen and Dan Halperin. Ten problems in geobotics. *CoRR*, abs/2408.12657, 2024. URL: https://doi.org/10.48550/arXiv.2408.12657, arXiv:2408.12657, doi:10.48550/ARXIV.2408.12657.

2 H. G. Eggleston. *Convexity*. Cambridge Tracts in Mathematics and Mathematical Physics, No. 47. Cambridge University Press, New York, 1958.

3 Guillermo Esteban, Dan Halperin, Víctor Ruíz, Vera Sacristán, and Rodrigo I. Silveira. Shortest coordinated motion for square robots. In Pat Morin and Subhash Suri, editors, *Algorithms and Data Structures - 18th International Symposium (WADS 2023)*, pages 430–443, Cham, 2023. Springer Nature Switzerland.

4 Sándor P. Fekete, Ramin Kosfeld, Peter Kramer, Jonas Neutzner, Christian Rieck, and Christian Scheffer. Coordinated motion planning: Multi-agent path finding in a densely packed, bounded domain. In Julián Mestre and Anthony Wirth, editors, *35th International Symposium on Algorithms and Computation, ISAAC 2024, December 8-11, 2024, Sydney, Australia*, volume 322 of *LIPIcs*, pages 29:1–29:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. URL: https://doi.org/10.4230/LIPIcs.ISAAC.2024.29, doi:10.4230/LIPICS.ISAAC.2024.29.

5 Christian Icking, Günter Rote, Emo Welzl, and Chee-Keng Yap. Shortest paths for line segments. *Algorithmica*, 10(2-4):182–200, 1993. URL: http://dx.doi.org/10.1007/BF01891839, doi:10.1007/BF01891839.

6 David Kirkpatrick and Paul Liu. Minimum-length coordinated motions for two convex centrally-symmetric robots. 2025. arXiv:2503.02010.

7 David G. Kirkpatrick and Paul Liu. Characterizing minimum-length coordinated motions for two discs. In Thomas C. Shermer, editor, *Proceedings of the 28th Canadian Conference on Computational Geometry, CCCG 2016, August 3-5, 2016, Simon Fraser University, Vancouver, British Columbia, Canada*, pages 252–259. Simon Fraser University, Vancouver, British Columbia, Canada, 2016.

8 Paul Liu. *Characterizing minimum-length coordinated motions for two disks.* PhD thesis, University of British Columbia, 2017. URL: https://open.library.ubc.ca/collections/ubctheses/24/items/1.0345619, doi:http://dx.doi.org/10.14288/1.0345619.

Strictly output sensitive color frequency reporting

Erwin Glazenburg[1] **and Frank Staals**[2]

1 **Utrecht University**
`e.p.glazenburg@uu.nl`
2 **Utrecht University**
`f.staals@uu.nl`

Abstract

Let P be a set of colored points. Given a query rectangle Q, we wish to report all k colors of points contained in Q together with their frequencies, in time strictly linear in the output complexity k. This appears to be difficult, so we study two relaxations: (1) the offline or batched version where we answer m queries at once, and (2) a version where we allow only restricted queries.

1 Introduction

Let P be a set of n points in $\mathbb{R}^2$, each of which has a color from the set $\{1..\phi\}$, and let P_c denote the subset of points of color c. Motivated by chromatic nearest neighbor queries [13] we wish to store P so that given an axis aligned query rectangle Q we can efficiently report the colors of the points appearing in the query range, together with their frequency. That is, we wish to report a set of k color, frequency pairs (c, f) so that $|P_c \cap Q| = f > 0$ and the sum of the frequencies is $|P \cap Q|$. See Figure 1. This is also known by the rather non-descript name of *type-2* color counting [9]. Our main goal is to obtain a linear space data structure achieving a *strictly output sensitive* query time of the form $O(q(n) + k)$, i.e. a query time strictly linear in the output size k (and whose dependency $q(n)$ on n is of course also sublinear, and preferably even (poly-)logarithmic). We work in the pointer-machine model of computation.

Related work. Color frequency counting is a form of *generalized intersection searching* [8, 9, 10]. In 1D, both the problem of reporting the color frequencies of points within a query interval [2] and the problem of reporting the color frequencies of intervals containing a query point [9] have been solved since 1995 with datastructures of size $O(n)$ and query time $O(\log n + k)$. In 2D there exists a simple datastructure of size $O(n \log n)$ with query time $O(\log n + k \log n)$: we first report all intersected colors in $O(\log n + k)$ time using a colored range reporting query [12], and then for all k colors separately perform a standard uncolored range counting query in $O(\log n)$ time each using 2D range trees [4]. On the other hand, Gupta et al. [8] (Theorem 1.11) show how to obtain $O(\log n + k)$ query time using $O(n^{1+\varepsilon})$ space, where $\varepsilon > 0$ is some arbitrarily small constant (their result is for color reporting, but also applies to color frequency reporting); this works in any constant dimension d, not just in

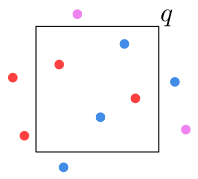


Figure 1 Some colored points and a query rectangle q, with color-frequencies ((red, 2), (blue, 2)).

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

2D. Note that neither of these approaches achieve our goal of using linear space with strictly output sensitive query time.

In the word-RAM model, some progress has been made recently. In 2020, Chan et al. [3] give a datastructure of size $O(n \log^{1+\varepsilon} n)$ with query time $O(\frac{\log n}{\log \log n} + k \log \log n)$. In 2023, Afshani et al. [1] solve approximate color frequency counting with an *additive* error ε for dominance queries with $O(n)$ space and $O(\log n + \varepsilon^{-1})$ query time. We are not aware of similar results in the pointer-machine model.

In external memory with block size B, Ganguly et al. [7] give a datastructure of size $O(n)$ that answers *constant-factor approximate* frequency reporting queries in $O(\log_B n + \log^* B + k/B)$ time.

Challenges & Results. We take some steps towards our goal of strict output sensitivity using linear space. This turns out to be challenging: it is unclear how to achieve even e.g. $O(n^{0.99} + k)$ query time. Most data structures follow a divide-and-conquer approach that partitions the space (and thereby the input point set). However, this may cause the points $P_c \cap Q$ of color c to be stored in many, say $g(n)$, different places in the data structure. This means that we have to aggregate their frequencies at query time. If this happens for $\Omega(k)$ colors, we get an $\Omega(kg(n))$ cost in the query time. If we instead store the points per color, we may have to spend $\Theta(\log n)$ time per color to actually count the subset that lies in Q. We present progress on two restricted variants of the problem.

- The offline problem. If we are given m queries in a batch, we can answer all of them in $O(\sqrt{m} \log m(m + n \log^3 n)) + K)$ time using $O(n)$ space, where K is the total output complexity of all queries. See Section 3. Note that e.g. for $m = \Theta(n)$ we thus essentially obtain $O(\sqrt{n} \log^4 n + k)$ time per query. This improves over the naive near-linear space solution (with query time $O(\log n + k \log n))$ by a factor $\log n$ when the average output complexity of a query is $\Omega(\sqrt{n} \log^4 n)$.
- In the above approach we encounter a batch of dominance frequency reporting queries where the point defining the query range is separated from P by an $(x, -y)$-monotone curve. We show that we can also solve the online version of this problem efficiently in Section 4, by reducing the problem to frequency reporting for interval $\times$ interval intersection.

2 Preliminaries

For two (corner) points p and q, we say p is *dominated* by q, written $p \prec q$, iff p lies in the bottom-left quadrant of q. If neither $p \prec q$ nor $q \prec p$, then p and q are *incomparable.*

We focus on dominance frequency reporting, where the query region is a *quadrant* or *2-sided* rectangle $(-\infty, x] \times (-\infty, y]$ defined by its top-right corner (x, y). See Figure 2. Given a data structure for dominance queries we can then also efficiently answer queries in which the query range is a *3-sided*, or a *4-sided* rectangle $[x_1, x_2] \times [y_1, y_2]$.

▶ **Theorem 2.1.** *Assume we are given a datastructure for dominance frequency reporting using $S(n)$ space with $Q(n,k)$ query time (with $Q(n,k)/\log n$ non-decreasing), where the colors are reported in a fixed order. We can then create a datastructure for frequency reporting with 3-sided rectangles in $O(Q(n,k))$ time using $O(S(n) \log n)$ space, or with 4-sided rectangles in $O(Q(n,k))$ time using $O(S(n) \log^2 n)$ space.*

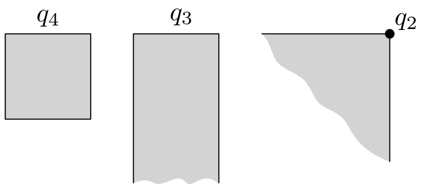


Figure 2 A four-, three-, and two-sided rectangle q_4, q_3, and q_2.

3 Offline algorithm for dominance frequency reporting

We first consider the offline version of the dominance frequency reporting problem in which we wish to answer a set $Q = \{q_1, .., q_m\}$ of m dominance frequency reporting queries on P. We show how to answer all m queries in $O(\sqrt{m} \log m(m + n \log n) + K)$ time, where $K = \sum_i k_i$, and k_i is the output complexity of query q_i.

First note that, if we have $O(n^{1+\varepsilon})$ space available (for some $\varepsilon > 0$), we can build the $O(n^{1+\varepsilon})$ space datastructure from Gupta et al. [8] and answer all queries in $O(\log n + k)$ time each. In this setting we thus assume to have only linear space available.

Let $Q' = q_1, .., q_{m'} \subseteq Q$ be an ordered set of queries. If $q_i \prec q_j$ for all $i < j$, then we say Q' is a *dominating sequence*. If neither $q_i \prec q_j$ nor $q_j \prec q_i$ for all pairs of queries q_i and q_j, and Q' is sorted by x-coordinate, then we say Q' is an *incomparable sequence*. See Figure 3. This extra structure allows us to solve all ℓ queries in Q' efficiently:

▶ **Lemma 3.1.** *We can answer all queries in a dominating or incomparable sequence Q' of length m' in $O(n \log n + m' + K')$ time using $O(n)$ space, where $K' = \sum_i k_i$ is the total output size of Q'.*

Proof sketch. See Figure 3. For a dominating sequence we know that, for all $i < j$, all points contained in q_i are also contained in q_j, meaning the answer only 'grows' as we go from q_1 to $q_{m'}$. As such we can sweep a point q from the bottom-left to the top-right, starting at $(-\infty, -\infty)$ and passing through all queries in order. We maintain the color frequencies of the set of points $P_{bl} \subseteq P$ to the bottom-left of q, and each time we reach a query q_i we simply output this answer in $O(m' + K')$ total time. Each point is added to P_{bl} once, in $O(\log n)$ time each, so in $O(n \log n)$ total time.

We handle an incomparable sequence similarly. Observe that an incomparable sequence forms a staircase. We can sweep a point q along the staircase from top-left towards bottom-right, maintaining the set of points P_{bl} in its bottom-left quadrant. We output the current answer each time we reach a query in $O(m' + K')$ total time. Each point is added to and removed from P_{bl} at most once, in $O(n \log n)$ total time. ◀

The crux of the algorithm is that in any set of m queries there exists either a dominating sequence or a incomparable sequence of size at least $\sqrt{m}$; this is known as the Erdös-Szekeres theorem [5]. We can find such a large sequence in $O(m \log m)$ time [6, 11]. For the purpose of self-containment we give a simple geometric proof of these two (known) facts FRANK: for the case of dominance ranges below.

▶ **Lemma 3.2** (Erdös-Szekeres theorem). *Given a set Q of m queries, we can either find (i) a set $Q' \subseteq Q$ of $\sqrt{m}$ dominating queries, or (ii) a set $Q' \subseteq Q$ of $\sqrt{m}$ incomparable queries, in $O(m \log m)$ time.*

Proof. For ease of description we insert a query q_0 that is dominated by all other queries.

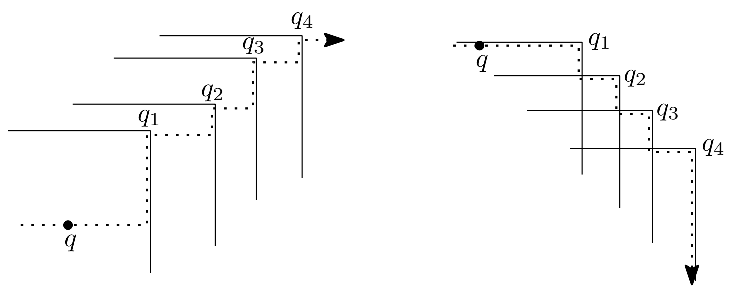


Figure 3 A dominating sequence and an incomparable sequence, with the path of q shown.

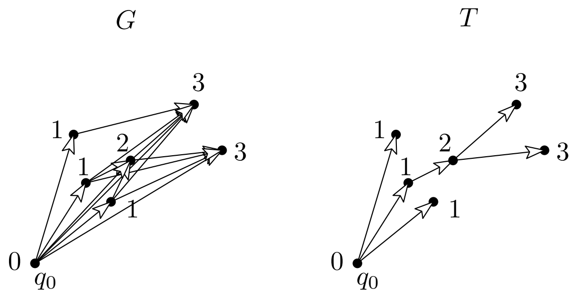


Figure 4 The graphs G and T on a set of queries, with the level of each query.

Consider the dominance graph $G = (Q, A)$ where there is a directed arc $(q_i, q_j) \in A$ if and only if $q_i \prec q_j$. Let the level $\ell(q_i)$ of a query q_i be the length of a longest path from q_0 to q_i in G. See Figure 4.

Suppose there exists a query q_i with $\ell(q_i) \geq \sqrt{m}$. Then there is a path of length $\sqrt{m}$ in G, and this path corresponds to a dominating sequence of length at least $\sqrt{m}$, which would establish (i). Otherwise, all queries have level less than $\sqrt{m}$. Then by the pigeonhole principle there must be a level with at least $m/\sqrt{m} = \sqrt{m}$ queries in it. All queries at the same level are incomparable, since if we have two queries q_i and q_j with $q_i \prec q_j$, then q_j must be an ancestor of q_i in G, so $\ell(q_j) > \ell(q_i)$. Hence, establishing (ii).

We have now shown that either (i) or (ii) holds, but graph G can have $O(m^2)$ arcs and will thus take too long to construct. Instead we construct a subgraph T of G with only $O(m)$ arcs that preserves a longest path from q_0 to—and thus the level of—each query.

Consider a query $q_j \neq q_0$ in G of level $\ell(q_j) = t$, and consider all arcs (q_i, q_j) pointing to q_j for which $\ell(q_i) = t - 1$. Note that there must be at least one such arc, otherwise q_j can not have level t. We insert only the arc coming from the leftmost such q_i into T. The level of each query in T is the same as in G, which we can see by induction on the level t. As a base case, q_0 clearly still has level 0. For the inductive step, our induction hypothesis is that all queries of level t have the same level as in G. Each query q_j of level $t + 1$ in G has exactly one incoming arc in T coming from a node of level t by definition of T; therefore, q_j also has level $t + 1$ in T, finishing the induction. We can construct T in $O(m \log m)$ time by sweeping a horizontal line upwards, computing the level and incoming arc of all queries below the sweepline. ◀

We will repeatedly apply Lemma 3.2 to find the longest remaining dominating or incomparable sequence of queries, and answer them using Lemma 3.1. We work in rounds. In each round $r = 1, 2, ..$ we get a set of m_r queries, with $m_1 = m$, and we find a set Q_r of $\sqrt{m_r}$ dominating or incomparable queries in $O(m_r \log m_r)$ time. We answer these using

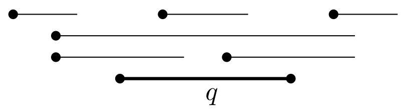


Figure 5 The six possible intervals w.r.t. q, with their left endpoint highlighted.

Lemma 3.1 in $O(n \log n + \sqrt{m_r} + K_r)$ time (where K_r is the output size of Q_r). Each round r thus takes thus $O(m_r \log m_r + n \log n + K_r)$ time. In the next round $r+1$ we have $m_{r+1} = m_r - \sqrt{m_r}$ queries left. Ignoring the output sensitive K, the total runtime can thus be written as $T(m, n) = T(m - \sqrt{m}, n) + O(m \log m + n \log n)$, which solves to $T(m, n) = O(\sqrt{m} \log m(m + n \log n))$.

▶ **Theorem 3.3.** *We can answer a set Q of m dominance frequency reporting queries in $O(\sqrt{m} \log m(m + n \log n)) + K)$ time, using $O(n)$ space.*

Similar to Theorem 2.1 we can generalize this result to (4-sided) rectangles:

▶ **Theorem 3.4.** *We can answer a set Q of m rectangle frequency reporting queries in s time, using $O(n \log^2 n)$ space.*

4 Restricted queries

In Lemma 3.1 we saw an efficient algorithm to answer a batch of incomparable queries, which form a staircase. Note that we could ignore all points above the staircase, as they are not contained in any of the queries. So, all relevant points lie below the staircase, and in other words the queries are 'separated' from the points P. More precisely, there exists an $(x, -y)$-monotone curve C such that all points P lie below C, and all queries Q lie above C (e.g. the staircase itself translated to the bottom-left a tiny amount). In this section we show that this scenario does not only allow for a fast algorithm in the offline setting, but also in the online setting, where we are given an $(x, -y)$-monotone curve C such that all points P lie below C, and we wish to answer queries that lie above C. We call this the *curve-restricted* version of the dominance frequency reporting problem.

In order to answer these curve-restricted queries, we first show how to answer 1D interval intersection frequency reporting queries. Here we have a set of n colored 1D intervals, and given a query interval q wish to report the color frequencies of all intervals intersecting q.

▶ **Lemma 4.1.** *We can build a datastructure of size $O(n)$ to answer interval intersection frequency reporting queries in $O(\log n + k)$ time.*

Proof. Let $q = (q_l, q_r)$ be a query interval. Combinatorically, there are 6 different types of intervals with respect to q, as shown in Figure 5, depending on if the start and endpoint is before, inside, or after q. We wish to count exactly (1) the intervals whose left endpoint lies inside q, and (2) the intervals that contain q's left endpoint. Note that these two sets are non-overlapping. To this end we build (1) a range frequency reporting datastructure on the left endpoints of all intervals [2], and (2) an interval-stabbing frequency reporting datastructure on the intervals themselves [9], both using $O(n)$ space. We query both datastructures once per query in $O(\log n + k)$ time, leading to the claimed result. ◀

We can use this to solve the restricted dominance frequency reporting problem:

▶ **Lemma 4.2.** *We can build a datastructure for the restricted dominance frequency reporting problem using $O(n \log n)$ space, with $O(\log n + k)$ query time.*

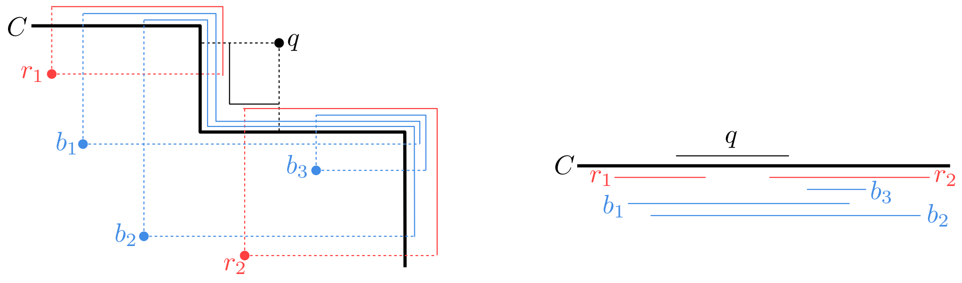


Figure 6 Left: a set of points below a curve C (drawn axis-aligned for simplicity of drawing) and a query above it, with their respective intervals drawn. Right: the intervals drawn in 1D.

Proof. We show that a query in this problem can be reduced to a 1D interval intersection problem; the claimed time and space then follow from Lemma 4.1.

See Figure 6. For each point p, consider the interval I_p of the curve C contained in its top-right quadrant. Similarly, for a query q consider the interval I_q of the curve C contained in its bottom-left quadrant. A query q contains a point p if and only if I_q and I_p overlap: iff I_q and I_p overlap then there is some point s on C such that $p \prec s \prec q$, so q contains p. Therefore we can use an interval intersection datastructure, built on the intervals I_p and queried with intervals I_q, to solve this problem. We ignore here the time required to actually compute I_q; this may depend on the actual curve C chosen, but for example for an axis-aligned staircase this can be computed in $O(\log n)$ time. ◀

References

1 Peyman Afshani, Pingan Cheng, Aniket Basu Roy, and Zhewei Wei. On range summary queries. *arXiv preprint arXiv:2305.03180*, 2023.

2 Panayiotis Bozanis, Nectarios Kitsios, Christos Makris, and Athanasios Tsakalidis. New upper bounds for generalized intersection searching problems. In *Automata, Languages and Programming: 22nd International Colloquium, ICALP 95 Szeged, Hungary, July 10–14, 1995 Proceedings 22*, pages 464–474. Springer, 1995.

3 Timothy M Chan, Qizheng He, and Yakov Nekrich. Further results on colored range searching. *arXiv preprint arXiv:2003.11604*, 2020.

4 Mark De Berg. *Computational geometry: algorithms and applications*. Springer Science & Business Media, 2000.

5 Paul Erdös and George Szekeres. A combinatorial problem in geometry. *Compositio mathematica*, 2:463–470, 1935.

6 Michael L Fredman. On computing the length of longest increasing subsequences. *Discrete Mathematics*, 11(1):29–35, 1975.

7 Arnab Ganguly, J Ian Munro, Yakov Nekrich, Rahul Shah, and Sharma V Thankachan. Categorical range reporting with frequencies. In *22nd International Conference on Database Theory (ICDT 2019)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2019.

8 Prosenjit Gupta, Ravi Janardan, Saladi Rahul, and Michiel Smid. Computational geometry: Generalized (or colored) intersection searching. In *Handbook of Data Structures and Applications*, pages 1043–1058. Chapman and Hall/CRC, 2018.

9 Prosenjit Gupta, Ravi Janardan, and Michiel Smid. Further results on generalized intersection searching problems: counting, reporting, and dynamization. *Journal of Algorithms*, 19(2):282–317, 1995.

10 Ravi Janardan and Mario Lopez. Generalized intersection searching problems. *International Journal of Computational Geometry & Applications*, 3(01):39–69, 1993.

11 Jiří Matoušek and Emo Welzl. Good splitters for counting points in triangles. In *Proceedings of the fifth annual symposium on Computational geometry*, pages 124–130, 1989.

12 Qingmin Shi and Joseph JáJá. Optimal and near-optimal algorithms for generalized intersection reporting on pointer machines. *Information Processing Letters*, 95(3):382–388, 2005.

13 Thijs van der Horst, Maarten Löffler, and Frank Staals. Chromatic k-nearest neighbor queries. *arXiv preprint arXiv:2205.00277*, 2022.

Visualizing Hypergraphs as Metro Maps: Drawing Paths with Few Bends in Trees and Cacti [*]

Sabine Cornelsen[1], Henry Förster[2], Siddharth Gupta[3], Stephen Kobourov[2], and Johannes Zink[2]

1 **University of Konstanz, Germany**
2 **TU Munich, Heilbronn, Germany**
3 **BITS Pilani, K K Birla Goa Campus, India**

Abstract

A *path-based support* is a graph together with a set of paths. We consider the problem of constructing straight-line drawings of path-based tree or cactus supports that (i) minimize the sum of the number of bends on all paths, (ii) minimize the *curve complexity*, i.e., the maximum number of bends on any path, or (iii) maximize the number of 0-bend paths, then the number of 1-bend paths, etc.

1 Introduction

A *hypergraph* $H = (V, A)$ is a set V of vertices and a set A of subsets of V, called hyperedges. Hypergraphs can be visualized as node-link bipartite graph drawings with vertices $V \cup A$, as *Hasse diagrams* (upward drawing of the transitive reduction of the inclusion relation between hyperedges), or as Euler diagrams (enclosing the vertices of each hyperedge by a simple closed curve). We consider the metro map metaphor, where each hyperedge is visualized with a metro line along which the vertices in the hyperedge are the stations; see Fig. 1. As each metro line is a path p_h for a hyperedge h, the union of the paths is a *path-based support* of the hypergraph $H = (V, A)$: a *support graph* $G = (V, E)$ and a set $\mathcal{P} = \{p_h; h \in A\}$ of paths of G such that p_h contains exactly the vertices of h. Moreover, each edge of G is contained in at least one path of $\mathcal{P}$. We say that G is a *planar path-based support*, a *path-based tree support*, or a *path-based cactus support* if G is planar, a tree, or a cactus[1], respectively. A special case are *linear hypergraphs/supports* where any two hyperedges/paths share at most one vertex.

We study how to draw a support graph G so that the number of bends per path or the total number of bends on all paths is small. We focus on straight-line drawings where paths can only bend at vertices. The *curve complexity* of a drawing of a path-based support is the minimum b such that each path has at most b bends. The *(planar) curve complexity* of a path-based support is the minimum curve complexity over all its (planar) straight-line drawings. If in a planar embedding of a linear path-based support $(G, \mathcal{P})$ no two paths touch, i.e., they are either disjoint or properly intersect, the question if G is drawable such that no path has a bend corresponds to the $\exists\mathbb{R}$-hard problem of pseudo-segment stretchability [19].

Related Work. Visualizing hypergraphs with the metro map metaphor has practical applications [13] and gives rise to theoretical considerations [11]. It is NP-complete to decide if there is a planar path-based support and NP-hard to find a path-based support with the minimum number of edges [5], however a path-based tree support [20] and a (not necessarily path-based) cactus support [4] can be constructed in polynomial time, if they exist. More

[*] This research began at the Graph and Network Visualization Workshop 2024 (GNV'24) in Heiligkreuztal.
[1] A *cactus* is a connected graph where each edge is contained in at most one cycle.

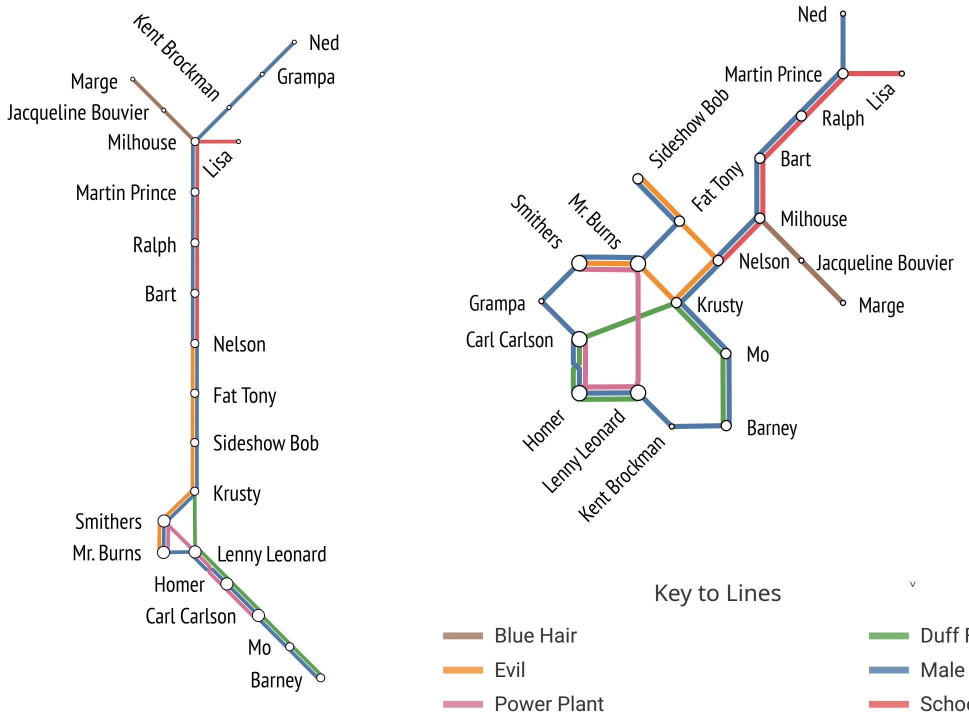


Figure 1 Different representations for the same hypergraph, from metrosets.ac.tuwien.ac.at [13].

work on computing planar supports can be found in [6, 7, 14, 15, 16]. Given a path-based support, the problem of embedding the support graph has also been studied: with spring embedders [12], mixed-integer programming [18], simulated annealing [3], and other techniques [1, 2, 10, 12, 17, 18]. Dobler et al. [8, 9] considered the problem of drawing hyperedges as lines or line-segments where the order of the vertices of the hyperedges is not given.

Our Contribution. In Section 2, we show that the (planar) curve complexity can be efficiently determined for linear path-based cactus supports. In Section 3 we show that the corresponding problem is NP-hard even for restrictive non-linear path-based tree-supports, whereas the total number of bends can be efficiently minimized. We conclude with the parameterized complexity of determining the planar curve complexity, minimizing the sum of bends, maximizing the number of 0-bend, then 1-bend, etc. paths; see Fig. 2 for a comparison of the objectives.

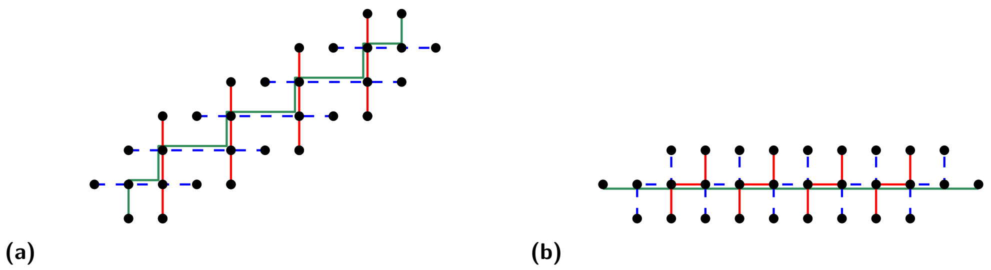
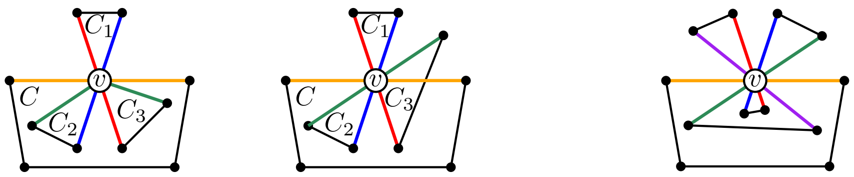


Figure 2 If we first draw as many paths with zero bends as possible, then there can be a path with $\Theta(n)$ bends in a caterpillar with n vertices (a), even though the curve complexity is 2 (b).



(a) planar curve complexity 1, curve complexity 0 **(b)** planar curve complexity 0

Figure 3 Linear path-based cactus supports. Paths of length at least two are indicated by colors.

2 Linear Hypergraphs

Two adjacent edges are *aligned* in a drawing of a support if they are contained in one line and disjoint except for their common end point. Any alignment requirements for a tree can be realized in a planar way. Thus, the planar curve complexity of a tree equals its curve complexity. However, this is not true for linear path-based cactus supports (Fig. 3a).

▶ **Theorem 2.1.** *The curve complexity of a linear path-based cactus support is 0.*

Proof Sketch. Consider bridges as cycles of length two. We show by induction on the number of cycles that there is a drawing in which all cycles are drawn convex and in which two incident edges are aligned if and only if they are contained in the same path. This is possible, since in a linear support every cycle contains edges of at least three different paths, and, thus, can bend at least three times. ◀

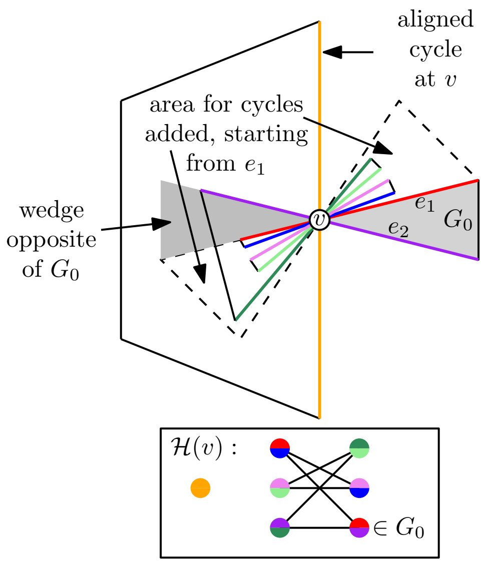
Cycle C of a path-based support is *aligned* at a vertex v if its two edges incident to v belong to the same path. For vertex v of a path-based cactus support, consider the *constraint graph* $\mathcal{H}(v)$ that has a node for each cycle containing v and, for each path containing v and an edge from each of the two cycles, has an edge between the corresponding two nodes. E.g., in Figure 3a, graph $\mathcal{H}(v)$ contains triangle $\langle C_1, C_2, C_3\rangle$, while in Figure 3b it contains a 4-cycle.

▶ **Theorem 2.2.** *The planar curve complexity of a linear path-based cactus support is 0 if and only if (i) at most one cycle at any vertex is aligned and (ii) if a cycle C is aligned at vertex v, then the constraint graph of v is bipartite. This can be determined in linear time.*

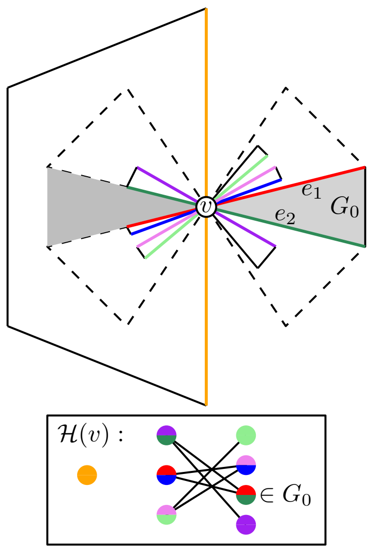
Proof Sketch. If two cycles are aligned, then the two cycles must cross in a drawing without bends. If a cycle C is aligned at a vertex v, then the cycles of any path in $\mathcal{H}(v)$ must alternate between the interior and the exterior of C in a planar drawing without bending paths. Thus, $\mathcal{H}(v)$ is bipartite. Assume that Conditions (i)–(ii) are fulfilled for all vertices. By induction on the number of cut vertices, we show that there is a planar drawing in which all cycles are drawn convex and two incident edges are aligned if and only if they are contained in the same path. A *component* of a vertex v is the subgraph of G induced by a connected component of $G - v$ together with v. Choose a cut vertex v such that all but at most one component G_0 of v consists of a simple cycle or an edge. Starting from a drawing of G_0, we add the other components along maximal paths in $\mathcal{H}(v)$; see Fig. 4. ◀

▶ **Corollary 2.3.** *The planar curve complexity of a linear path-based support is 0 if the underlying support graph is a tree and at most 1 if the underlying support graph is a cactus.*

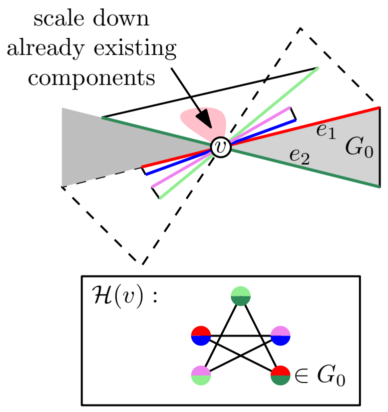
Proof. The conditions in Theorem 2.2 are trivially fulfilled for a tree, since there are no cycles. Follow the construction in the proof of Theorem 2.2. When inserting new components, break all paths at v unless the path contains an edge of G_0. This way the conditions in Theorem 2.2 are fulfilled and the first drawn vertex of a path will be its only bend, if any. ◀



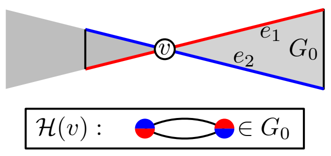
(a)



(b)



(c)



(d)

Figure 4 Linear path-based cactus supports admitting planar drawings with curve complexity 0. (a),(c) Even and odd cycle in $\mathcal{H}(v)$, resp. (b) Path in $\mathcal{H}(v)$ through cycle in G_0. (d) 2-cycle in $\mathcal{H}(v)$.

3 Not Necessarily Linear Hypergraphs

Different from linear hypergraphs, the curve complexity in trees and cacti is in general unbounded. The curve complexity of path-based tree supports is in $\Omega(\log n)$: in the complete binary tree with n vertices and a path from each leaf to the root, one path must bend at each inner vertex. The curve complexity of path-based cactus supports is in $\Omega(n)$: among the paths $P : \langle v_0, v_1, v_2, \dots, v_{n-1}\rangle$ and $Q : \langle v_0, v_2, v_4, \dots\rangle$, P bends at least $n/2 - 1$ times.

▶ **Theorem 3.1.** *It can be decided in polynomial time whether the (planar) curve complexity of a path-based cactus support $(G = (V, E), \mathcal{P})$ is 0.*

Proof. If there are three edges e, e_1, and e_2 incident to v and two paths in $\mathcal{P}$, one containing e and e_1 and the other containing e and e_2 then there is no drawing without bends. Otherwise we merge paths that share an edge into one path. Thus, in the obtained cactus support $(G = (V, E), \mathcal{P}')$ no two paths share an edge and it has (planar) curve complexity zero if and only if $(G = (V, E), \mathcal{P})$ does. If there are two paths in $\mathcal{P}'$ that intersect twice then the two paths would contain a cycle and, thus, cannot both be drawn on a straight line. Otherwise, the support is linear and we can apply Theorem 2.1 or Theorem 2.2, respectively. ◀

▶ **Theorem 3.2.** *Given a path-based tree support $(G = (V, E), \mathcal{P})$, an alignment of E, in which the sum of the bends in all paths is minimum, can be computed in polynomial time.*

Proof. For each vertex v, consider the complete graph $G(v)$ on the set $N(v)$ of neighbors of v. The weight of an edge $\{u, w\}$ of $G(v)$ is the number of paths in $\mathcal{P}$ that contain $\{u, v\}$ and $\{v, w\}$. Consider a maximum matching $M(v)$ on $G(v)$ and align two edges $\{u, v\}$ and $\{v, w\}$ if and only if $\{u, w\} \in M(v)$. These maximum matchings yield an optimum solution. ◀

▶ **Theorem 3.3.** *It can be decided in polynomial time whether the planar curve complexity of a path-based tree support $(G = (V, E), \mathcal{P})$ is 0 or 1.*

Proof. We use a 2-SAT formulation. There is a variable $x_{e_1e_2}$ for each pair e_1, e_2 of incident edges. We interpret $x_{e_1e_2}$ as true if and only if e_1 and e_2 are not aligned. For any three incident edges e_1, e_2, e_3 we need the consistency condition that whenever e_1 is aligned to e_2 then e_1 cannot be aligned to e_3, i.e., the clause $x_{e_1e_2} \vee x_{e_1e_3}$. If we insist on having no bends then we need for any two incident edges e_1, e_2 on the same path the condition $\neg x_{e_1e_2}$. If we aim for at most one bend per path then we add for any four edges e_1, e_2, e_3, e_4 on a path where e_1, e_2 and e_3, e_4 are incident the condition that at least one among the two pairs must be aligned, i.e., the clause $\neg x_{e_1e_2} \vee \neg x_{e_3e_4}$. This yields a 2-SAT formulation with $\sum_{v\in V} \deg(v)(\deg(v) - 1)$ variables and $\mathcal{O}(\sum_{p\in\mathcal{P}} |p|^2 + \sum_{v\in V} \deg^3(v))$ clauses. ◀

The curve complexity is at most two if the support graph is a caterpillar. Thus, the curve complexity of a path-based caterpillar support can be determined in polynomial time.

▶ **Theorem 3.4.** *It is NP-hard to decide whether the curve complexity of a path-based tree support is at most b even if (i) $b = 3$ and the diameter of the support graph is at most five or (ii) the maximum degree of the support graph is at most three.*

Proof Sketch. Given a 3-SAT formula Φ with n variables and m clauses, we construct a tree support adhering to Constraints (i) and (ii), resp. In the following, we give the details for (i), i.e., $b = 3$ and diameter five (see Fig. 5).

For each variable v_i, $i = 1, \dots, n$, there is a variable gadget that consists of a path $v_i, x_i, \neg v_i$. A clause gadget of a clause c_j, $j = 1, \dots, m$, is a perfect binary tree of height two

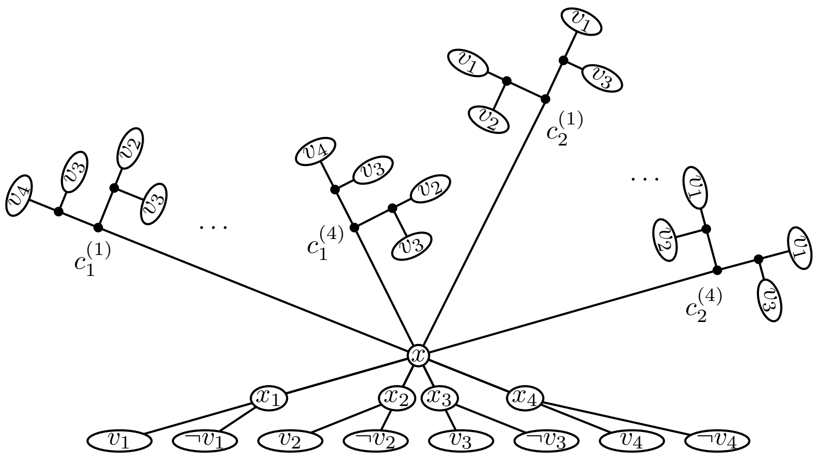


Figure 5 Graph corresponding to clauses $c_1 = v_2 \vee v_3 \vee v_4$ and $c_2 = v_1 \vee v_2 \vee v_3$. The drawing corresponds to a truth assignment in which v_1, v_3, and v_4 are true and v_2 is false.

rooted at a vertex labeled c_j whose leaves are labeled with the variables of c_j where one variable appears twice, once in each of the different subtrees rooted at the children of c_j. For each clause there are four copies of the clause gadget. Finally, there is a central vertex x that connects to each central vertex x_i, $i = 1, \ldots, n$, of the n variable gadgets as well as to each of the $4m$ central vertices labeled c_j, $j = 1, \ldots, m$, of the clause gadgets. See Fig. 5. The set of paths contains a path from any leaf labeled v_i of a clause gadget of a clause c_j to the leaf labeled v_i or $\neg v_i$ in the variable gadget of v_i, depending on whether v_i or $\neg v_i$ appears in c_j.

In a bend minimum drawing, the path from x to exactly one leaf of a clause gadget has two bends and 0 or 1 bend otherwise. There is at least one copy of each clause gadget such that each path ending there bends also in x. For each $i = 1, \ldots, n$, the edge $\{x, x_i\}$ is aligned with either $\{x_i, v_i\}$ or $\{x_i, \neg v_i\}$. Interpreting the former as v_i being set to true, then there is a satisfying truth assignment if and only if there is a drawing of curve complexity three.

For Case 2 we modify the NP-hardness proof: we construct a tree with $\mathcal{O}(nm)$ vertices and maximum degree 3 and a set of $\mathcal{O}(n^2m^2)$ paths that admits a drawing with $b = 2 \cdot (\lceil \log_2 n \rceil + \lceil \log_2 m \rceil + 1)$ bends per path if and only if Φ has a satisfying truth assignment. See Fig. 6. ◀

▶ **Theorem 3.5.** *The (planar) curve complexity for a path-based tree support $(G, \mathcal{P})$ is FPT parameterized by $|\mathcal{P}|$; there is a kernel with $\mathcal{O}(|\mathcal{P}|)$ vertices.*

Proof. For each leaf v, there is a path that ends in v, thus, the number of leaves is at most $2|\mathcal{P}|$. Shrink all vertices of degree 2, extending paths ending there. Now the number of inner vertices is less than the number of leaves, i.e. the size of the resulting tree is less than $4|\mathcal{P}|$. ◀

▶ **Theorem 3.6.** *The (planar) curve complexity for path-based tree supports is FPT parameterized by the vertex cover number k; there is a kernel with $\mathcal{O}(k^2)$ vertices and $\mathcal{O}(k^2)$ paths.*

Proof Sketch. Let $(G, \mathcal{P})$ be a path-based tree support. After removing all leaves of G, the resulting graph G' has at most $2k - 1$ vertices and at most $\binom{2k-1}{2}$ paths. Let $e = \{v, v'\}$ be an edge of G'. Let u, w be two leaves of G that were adjacent to v such that $\mathcal{P}$ contains a path through e and u or w, respectively. Then at least one of the two paths must bend at v. Moreover, if at least three paths share their central part in G' and extend on both ends to distinct leaves or if four paths extend to a pair of leaves on both ends, at least one of these

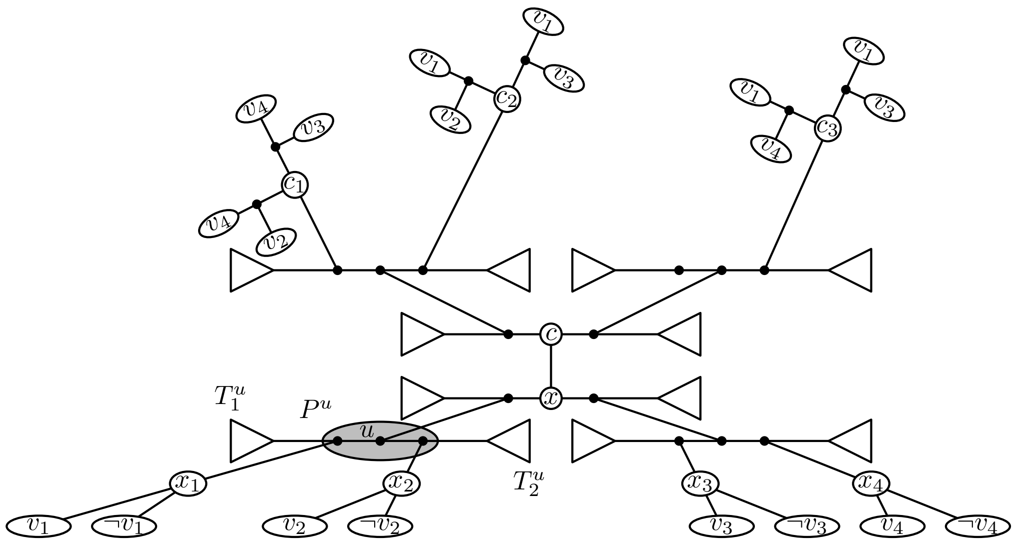


Figure 6 NP-hardness construction with maximum degree 3. Triangles indicate perfect binary trees of height $\lceil \log_2 n \rceil + \lceil \log_2 m \rceil + 1$ for n variables and m clauses; here $n = 4$ and $m = 3$. These perfect binary trees come in pairs T_1^u, T_2^u with paths between all leaves in T_1^u and all leaves in T_2^u.

paths has two bends at its leaves. It suffices to maintain at most six paths of $\mathcal{P}$ with the same part in G' in order to obtain a support with the same curve complexity as $(G, \mathcal{P})$. ◀

▶ **Theorem 3.7.** *The (planar) curve complexity for a path-based tree support is XP parameterized by the maximum number k of paths in $\mathcal{P}$ passing through a vertex. Deciding if the (planar) curve complexity is at most b is FPT parameterized by $k + b$.*

Proof Sketch. We employ a bottom-up dynamic programming algorithm. For each vertex v, we consider feasible drawings of the subtree G_v rooted at v including the parent edge of v. For each such drawing, we store for each of the at most k paths passing through v the number of bends in G_v. To this end, we use a simple brute-force strategy, testing all possible combinations of alignments of edges incident to v that are contained in the same path with possible choices of subdrawings for the children of v. As this exhaustively explores the search space, the correctness of the algorithm follows. For the FPT (XP) runtime, observe that in each feasible drawing each path through vertex v has between 0 and b bends (b is trivially bounded by n) while we have to check at most $2k$ edges for alignment (2 edges per path). ◀

If in addition, for each allowed number β of bends, we keep track of the number n_β of paths that are entirely located within the already drawn subtree, we get the following.

▶ **Corollary 3.8.** *The problem to first maximize the number of 0-bend paths, then the number of 1-bend paths, then the number of 2-bends paths, and so on, for path-based tree supports is XP parameterized by $k + b$ where k is the maximum number of paths in $\mathcal{P}$ for which a single vertex is internal and b denotes the maximum number of bends allowed on each path. Moreover, there is an XP-algorithm parameterized by $k + b^*$ where b^* denotes the maximum number of bends for which we want to maximize the number.*

By additionally keeping track of the number of bends per cycle and applying Theorem 3.1, the dynamic program generalizes to cacti.

▶ **Corollary 3.9.** *Theorems 3.7 and 3.8 generalize to path-based cactus supports.*

4 Open Problems for Tree and Cactus Supports

(a) What is the complexity of deciding whether the curve complexity is two? (b) Is the curve complexity W[1]-hard if parameterized by the maximum number of paths through a vertex? (c) Is it NP-hard to maximize the number of paths with no bends?

References

1 Hannah Bast, Patrick Brosi, and Sabine Storandt. Metro maps on octilinear grid graphs. *Comput. Graph. Forum*, 39(3):357–367, 2020. `doi:10.1111/CGF.13986`.

2 Hannah Bast, Patrick Brosi, and Sabine Storandt. Metro maps on flexible base grids. In Erik Hoel, Dev Oliver, Raymond Chi-Wing Wong, and Ahmed Eldawy, editors, *Proceedings of the 17th International Symposium on Spatial and Temporal Databases, SSTD 2021*, pages 12–22. ACM, 2021. `doi:10.1145/3469830.3470899`.

3 Michael A. Bekos, D. J. C. Dekker, F. Frank, Wouter Meulemans, Peter J. Rodgers, André Schulz, and S. Wessel. Computing schematic layouts for spatial hypergraphs on concentric circles and grids. *Comput. Graph. Forum*, 41(6):316–335, 2022. `doi:10.1111/CGF.14497`.

4 Ulrik Brandes, Sabine Cornelsen, Barbara Pampel, and Arnaud Sallaberry. Blocks of hypergraphs - applied to hypergraphs and outerplanarity. In Costas S. Iliopoulos and William F. Smyth, editors, *Proceedings of the 21st International Workshop on Combinatorial Algorithms, IWOCA 2010*, volume 6460 of *Lecture Notes in Computer Science*, pages 201–211. Springer, 2010. `doi:10.1007/978-3-642-19222-7\_21`.

5 Ulrik Brandes, Sabine Cornelsen, Barbara Pampel, and Arnaud Sallaberry. Path-based supports for hypergraphs. *J. Discrete Algorithms*, 14:248–261, 2012. `doi:10.1016/J.JDA.2011.12.009`.

6 Kevin Buchin, Marc J. van Kreveld, Henk Meijer, Bettina Speckmann, and Kevin Verbeek. On planar supports for hypergraphs. *J. Graph Algorithms Appl.*, 15(4):533–549, 2011. `doi:10.7155/JGAA.00237`.

7 Thom Castermans, Mereke van Garderen, Wouter Meulemans, Martin Nöllenburg, and Xiaoru Yuan. Short plane supports for spatial hypergraphs. *J. Graph Algorithms Appl.*, 23(3):463–498, 2019. `doi:10.7155/JGAA.00499`.

8 Alexander Dobler, Stephen Kobourov, William J. Lenhart, Tamara Mchedlidze, Martin Nöllenburg, and Antonios Symvonis. Representing hypergraphs by point-line incidences. *CoRR*, arXiv:2411.13985, 2024. `arXiv:2411.13985`.

9 Alexander Dobler, Stephen Kobourov, William J. Lenhart, Tamara Mchedlidze, Martin Nöllenburg, and Antonios Symvonis. Representing hypergraphs by point-line incidences. In Michael A. Bekos and Charis Papadopoulos, editors, *Proceedings of the 40th European Workshop on Computational Geometry, EuroCG 2024*, 2024. URL: https://eurocg2024.math.uoi.gr/data/uploads/paper_39.pdf.

10 Martin Fink, Herman J. Haverkort, Martin Nöllenburg, Maxwell J. Roberts, Julian Schuhmann, and Alexander Wolff. Drawing metro maps using Bézier curves. In Walter Didimo and Maurizio Patrignani, editors, *Proceedings of the 20th International Symposium on Graph Drawing, GD 2012*, volume 7704 of *Lecture Notes in Computer Science*, pages 463–474. Springer, 2012. `doi:10.1007/978-3-642-36763-2\_41`.

11 Fabian Frank, Michael Kaufmann, Stephen Kobourov, Tamara Mchedlidze, Sergey Pupyrev, Torsten Ueckerdt, and Alexander Wolff. Using the metro-map metaphor for drawing hypergraphs. In Tomás Bures, Riccardo Dondi, Johann Gamper, Giovanna Guerrini, Tomasz Jurdzinski, Claus Pahl, Florian Sikora, and Prudence W. H. Wong, editors, *Proceedings of the 47th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2021*, volume 12607 of *Lecture Notes in Computer Science*, pages 361–372. Springer, 2021. `doi:10.1007/978-3-030-67731-2\_26`.

12 Seok-Hee Hong, Damian Merrick, and Hugo A. D. do Nascimento. The metro map layout problem. In Neville Churcher and Clare Churcher, editors, *Proceedings of the Australasian Symposium on Information Visualisation, InVis.au 2004*, volume 35 of *CRPIT*, pages 91–100. Australian Computer Society, 2004. URL: http://crpit.scem.westernsydney.edu.au/abstracts/CRPITV35Hong.html.

13 Ben Jacobsen, Markus Wallinger, Stephen Kobourov, and Martin Nöllenburg. Metrosets: Visualizing sets as metro maps. *IEEE Trans. Vis. Comput. Graph.*, 27(2):1257–1267, 2021. doi:10.1109/TVCG.2020.3030475.

14 David S. Johnson and Henry O. Pollak. Hypergraph planarity and the complexity of drawing Venn diagrams. *J. Graph Theory*, 11(3):309–325, 1987. doi:10.1002/JGT.3190110306.

15 Michael Kaufmann, Marc J. van Kreveld, and Bettina Speckmann. Subdivision drawings of hypergraphs. In Ioannis G. Tollis and Maurizio Patrignani, editors, *Proceedings of the 16th International Symposium on Graph Drawing,GD 2008*, volume 5417 of *Lecture Notes in Computer Science*, pages 396–407. Springer, 2008. doi:10.1007/978-3-642-00219-9_39.

16 Boris Klemz, Tamara Mchedlidze, and Martin Nöllenburg. Minimum tree supports for hypergraphs and low-concurrency euler diagrams. In R. Ravi and Inge Li Gørtz, editors, *Proceedings of the 14th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2014*, volume 8503 of *Lecture Notes in Computer Science*, pages 265–276. Springer, 2014. doi:10.1007/978-3-319-08404-6_23.

17 Soeren Nickel and Martin Nöllenburg. Towards data-driven multilinear metro maps. *CoRR*, abs/1904.03039, 2019. arXiv:1904.03039.

18 Martin Nöllenburg and Alexander Wolff. Drawing and labeling high-quality metro maps by mixed-integer programming. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):626–641, 2011. doi:10.1109/TVCG.2010.81.

19 Marcus Schaefer. Complexity of some geometric and topological problems. In David Eppstein and Emden R. Gansner, editors, *Proceedings of the 17th International Symposium of Graph Drawing, GD 2009*, volume 5849 of *Lecture Notes in Computer Science*, pages 334–344. Springer, 2009. doi:10.1007/978-3-642-11805-0_32.

20 R. Swaminathan and Donald K. Wagner. On the consecutive-retrieval problem. *SIAM J. Comput.*, 23(2):398–414, 1994. doi:10.1137/S0097539792235487.

Discrete Fréchet Distance is Also Hard in Unweighted Planar Graphs

Ivor van der Hoog, Eva Rotenberg, and Lasse Wulf

Technical University of Denmark, Lyngby, Denmark
{idjva,erot,lawu}@dtu.dk

Abstract

The Fréchet distance is a well-established distance measure in computer science. We show that the discrete Fréchet distance of two disjoint simple curves in an unweighted planar graph cannot be approximated better than a factor of 1.2 in subquadratic time, unless the Orthogonal Vector Hypothesis (OVH) fails. This improves upon a recent result of Driemel, van der Hoog and Rotenberg [SoCG22] both in terms of generality and approximation factor.

1 Introduction

The Fréchet distance is a widely studied measure in computational geometry that quantifies the similarity between two curves. It captures the intuitive notion of traversal, often analogized to a person and a dog walking along two distinct paths while maintaining a flexible leash. The (strong) Fréchet distance of the two paths equals the shortest length the leash can possibly have, taken over all ways which the person and the dog may traverse their path. The Fréchet distance has many applications, such as hand signature matching, motion analysis, and geographic information systems [3, 6, 8, 9, 11–15, 17]. Alt and Godau [2] show how to compute the discrete Fréchet distance between curves with n and m vertices each in $O(mn \log(n + m))$ time. Agarwal et al. [1] improve this to $O(nm(\log \log n)/ \log n)$ (assuming $n \geq m$).

Conditional lower bounds for the Fréchet distance. Bringmann [4] showed, conditioned on OVH, that no strongly subquadratic algorithm can exist to compute the Fréchet distance between curves in the plane. Bringmann and Mulzer [5] extended the lower bound for intersecting curves in $\mathbb{R}^1$. Buchin, Ophelders and Speckmann [7] showed that (assuming OVH) there can be no strongly subquadratic algorithm that computes anything better than a 3-approximation for pairwise disjoint planar curves in $\mathbb{R}^2$ and intersecting curves in $\mathbb{R}^1$.

Driemel, van der Hoog, and Rotenberg recently introduced the discrete Fréchet distance of paths in a graph [10]. They showed that for all fixed $0 < \delta < 1$, the discrete Fréchet distance of two disjoint simple paths of length n and m in an integer-weighted *planar* graph cannot be approximated better than a factor of 1.01 in $O((nm)^{1-\delta})$ time, unless the orthogonal vector hypothesis (OVH) fails (assuming $m = \Omega(n^\gamma)$ for some constant $\gamma > 0$). This lower bound is complemented by a classic Dynamic Program which, given a distance oracle over the graph, computes the discrete Fréchet distance in $O(nm)$ time. The lower bound in [10] applies only to weighted graphs and does therefore not exclude the existence of a fast algorithm on unweighted planar graphs. A closer inspection of their argument reveals that their proof cannot simply be adapted to the unweighted case by subdividing long edges often enough. In this case, new vertices get introduced to the graph and their arguments break down. In this extended abstract we provide a conditional lower bound that applies to unweighted planar graphs as well. We slightly improve the approximation factor.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

▶ **Theorem 1.1.** *Let P, Q be disjoint simple paths in an unweighted planar graph. Let $|P| = n$ and $|Q| = m = n^\gamma$ for some constant $\gamma > 0$. Then for all $0 < \delta < 1$ one cannot approximate the discrete Frèchet distance between P and Q with a factor better than 1.2 in $O((nm)^{1-\delta})$ time unless OVH fails.*

2 Preliminaries

Fréchet distance. We assume we have a simple path $P = (p_1, \dots, p_n)$ with n vertices and a simple path $Q = (q_1, \dots, q_m)$ with m vertices such that they are vertex-disjoint, and contained inside a graph G with N vertices and M edges. The discrete Fréchet distance $\mathrm{D}_{\mathcal{F}}(P, Q)$ is defined in [10] as follows: A sequence $(a_i, b_i)_{i \in [k]}$ of pairs of indices is called a monotone walk if for all $i \in [k-1]$ we have $a_{i+1} \in \{a_i, a_i + 1\}$ and $b_{i+1} \in \{b_i, b_i + 1\}$. Let F be a monotone walk from $(1, 1)$ to (n, m). The cost of a walk is the maximum of $\mathrm{dist}(p_i, q_j)$ over $(i, j) \in F$, where dist() denotes the usual distance in an unweighted graph. The discrete Fréchet distance is the minimum cost of a monotone walk from $(1, 1)$ to (n, m).

$$\mathrm{D}_{\mathcal{F}}(P, Q) = \min_F \mathrm{cost}(F) = \min_F \max_{(i,j) \in F} \mathrm{dist}(p_i, q_j).$$

Orthogonal vectors. The OVH is a popular hardness conjecture. Given two sets of vectors $U, V \subseteq \{0, 1\}^d$ of sizes $|U| = n$ and $|V| = m$, the *OV problem* is to find out if some vector of U is orthogonal to some vector of V, i.e. if we have $\sum_{i=1}^d u_i v_i = 0$ for some $u \in U$ and $v \in V$.

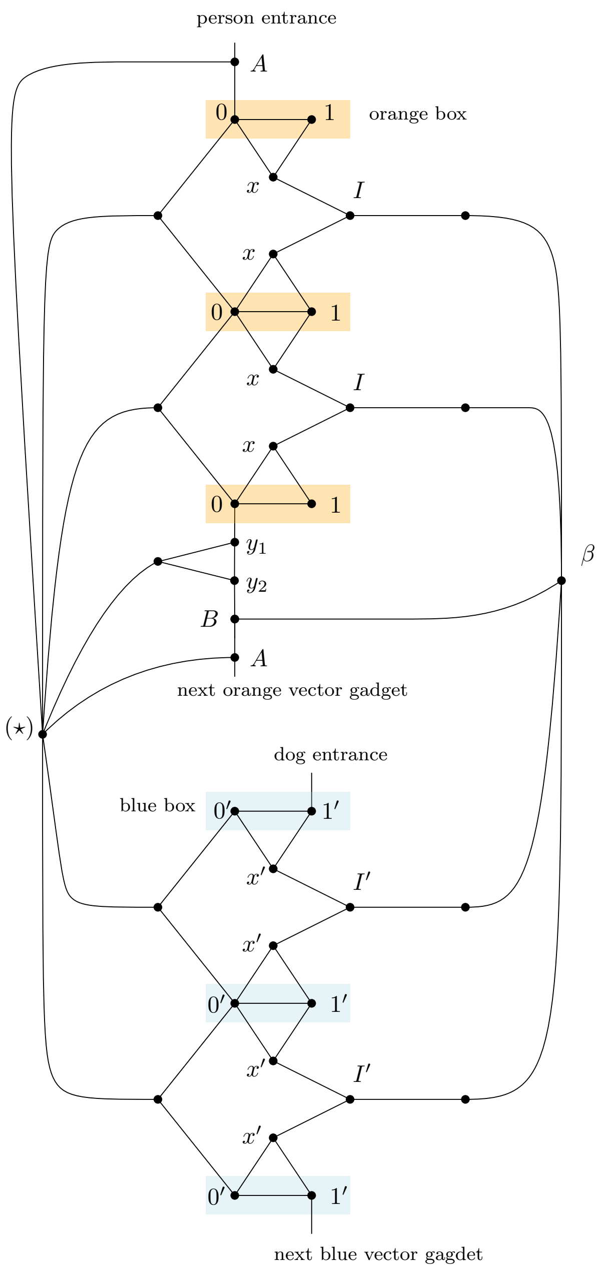
▶ **Definition 2.1** (Williams [16], see also Bringmann [4])**.** The orthogonal vector hypothesis states that for all $\delta > 0$, there exists constants $\omega > 0$ and $1 > \gamma > 0$ such that the OV problem for vectors of dimension $d = \omega \log n$ and $m = n^\gamma$ cannot be solved in $O((nm)^{1-\delta})$ time.

For space reasons, some proofs are omitted from this extended abstract.

3 Vector Gadgets

In this section we describe the lower bound construction based on the orthogonal vector hypothesis (OVH). The goal is to construct from some given OV instance in linear time a graph and two paths P, Q such that $\mathrm{D}_{\mathcal{F}}(P, Q) \leq 5$ if and only if the OV instance is a yes-instance and $\mathrm{D}_{\mathcal{F}}(P, Q) \geq 6$ otherwise. This suffices to prove Theorem 1.1. We start by describing two vector gadgets, one for the person, called the *orange* vector gadget, and one for the dog, called the *blue* vector gadget. Given two vectors $u, w \in \{0, 1\}^d$, we construct an orange gadget for u and a blue gadget for w. We assume w.l.o.g. that $u_1 = u_d = 0$ and $w_1 = w_d = 1$. This can be achieved w.l.o.g. by preprocessing the OVH instance. We prove that the gadgets have the following crucial property: Under the assumption that the person and dog are already positioned at the start of the gadget, the person and dog can traverse the gadgets maintaining a distance of 5 if and only if u and v are orthogonal.

Let us define the blue and orange vector gadgets for person and dog as in Figure 1. The construction comes with equivalence classes of vertices labelled either $0, 1, x, I, A, B, y_1, y_2$ in the orange gadgets, or $0', 1', x', I'$ in the blue gadgets, and two special vertices called β and $(\star)$. Let us call a pair of vertices $(0, 1)$ or $(0', 1')$ as highlighted in Figure 1 a *box*. The length of a vector gadget is the number of boxes in it. For a vector $u \in \{0, 1\}^d$, we can encode it inside an orange vector gadget of length d as a path $P(u)$ (see Figure 2). Let the term G_u denote the orange vector gadget corresponding to vector u.



	0	1	x	I
$0'$	4	5	5	6
$1'$	5	6	6	6
x'	5	6	6	5
I'	6	6	5	4

Figure 1 Vector gagdets for the person (orange) and the dog (blue) of length three, and a corresponding distance table.

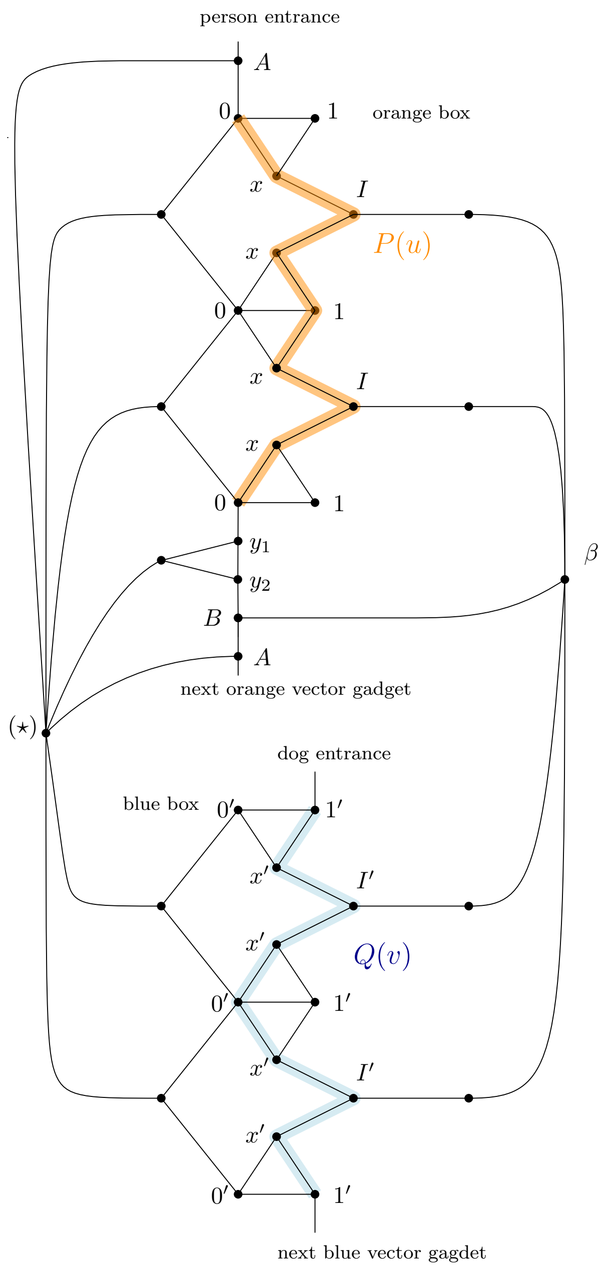


Figure 2 Example of encoding the vector $u = (0, 1, 0)$ in the orange gadget and the vector $v = (1, 0, 1)$ in the blue gadget. Note that the paths traverse exactly one vertex of each box, thereby encoding a binary vector.

The restriction of $P(u)$ to the orange boxes within the gadget G_u encodes the vector u. In general, we will define a path P through the graph that traverses each vector gadget once, such that for all orange vector gadgets G_u, the path P restricted to G_u equals $P(u)$. Likewise we can encode a vector $v \in \{0,1\}^d$ inside a blue vector gadget G_v of length d as a subpath $Q(v)$.

The pairwise distance between vertices of the blue and vertices of the orange gadgets is given in the table in Figure 1. We will later make sure that when we use these gadgets as part of a larger graph, that none of the distances described in the distance table changes. For the next lemma we require additional notation. If G_u is some orange vector gadget of length d, let $b_1(u), \dots, b_d(u)$ denote the d boxes in it. For some index $i \in \{1, \dots, d-1\}$, let us define $\mathrm{next}(b_i(u)) := b_{i+1}(u)$. For the last box in the gadget, the term $\mathrm{next}(b_d)(u)$ is undefined. Likewise, if G_v is some blue vector gadget of length d, we let $b'_1(v), \dots, b'_d(v)$ denote the d boxes in it. Similar as before we let $\mathrm{next}(b'_i(v)) := b'_{i+1}(v)$ for all $i \in \{1, \dots, d-1\}$ and $\mathrm{next}(b'_d(v))$ is undefined. Let $\mathcal{B}$ be the set of all orange boxes and $\mathcal{B}'$ be the set of all blue boxes, i.e.

$$\mathcal{B} := \bigcup_{u \in U} \{b_1(u), \dots, b_d(u)\}, \quad \mathcal{B}' := \bigcup_{v \in V} \{b'_1(v), \dots, b'_d(v)\},$$

where $U, V \subseteq \{0,1\}^d$ are the input sets of the original OV instance. Two boxes $b_i(u) \in \mathcal{B}$ and $b'_j(v) \in \mathcal{B}'$ are called *locally orthogonal*, if $u_i v_j = 0$, where u_i, v_j are the i-th and j-th component of the vectors u, v. This is equivalent to saying that at least one of the paths $P(u)$ and $Q(v)$ crosses a 0 or $0'$ in the box $b_i(u)$ or $b'_j(v)$.

▶ **Lemma 1.** Let $b \in \mathcal{B}, b' \in \mathcal{B}'$ be boxes such that both $\mathrm{next}(b), \mathrm{next}(b')$ are defined. Assume the person is sitting on the 0 or 1 in the box b, and the dog is sitting on the 0 or 1 in the box b'. If b, b' are locally orthogonal and $\mathrm{next}(b), \mathrm{next}(b')$ are locally orthogonal, then there is a way for the person and the dog to traverse their paths, while maintaining a distance of 5, such that after some time they are simultaneously sitting in $\mathrm{next}(b)$ and $\mathrm{next}(b')$.

In particular, this lemma implies the following: If the person and the dog are positioned such that the person is currently on the first vertex of $P(u)$ for some vector u, and the dog is on the first vertex of $Q(v)$ for some vector v, and (u, v) are orthogonal, then they can simultaneously traverse $P(u)$ and $Q(v)$, while maintaining a distance of 5.

Proof of Lemma 1. Since the boxes are locally orthogonal, either the person or the dog is currently sitting on a 0. We construct a traversal. First, the one sitting on a 0 stays put, while the other one makes a step. This is legal, since $\mathrm{dist}(0, x') = \mathrm{dist}(x, 0') = 5$. Now both make a step. This is legal, since $\mathrm{dist}(x, I') = \mathrm{dist}(I, x') = 5$. The one lagging behind makes a step, so that now both are on I and I'. Since the vectors are locally orthogonal at the next box, at least one of the person or the dog has a 0 in the next box. The being with a 0 in the next box makes a step, while the other stays put. This is again legal because $\mathrm{dist}(x, I') = \mathrm{dist}(I, x') = 5$. Both simultaneously make a step. Since the being with the 0 in the next box went first, they are now on a 0, and we have $\mathrm{dist}(0, x') = \mathrm{dist}(x, 0') = 5$. Finally, the being lagging behind makes the last step, so now both beings are inside the next box (which is legal since $\mathrm{dist}(0, 0'), \mathrm{dist}(0, 1'), \mathrm{dist}(1, 0') \leq 5$). ◀

The sufficient condition of Lemma 1 is accompanied by the following necessary condition. Given some orange box b, let us say the person is *adjacent to* b, if he is positioned either inside b, or at a vertex x immediately before or after b (but not at A or y_1, if b happens to be the first/last box). Given some blue box b', let us say the dog is *adjacent to* b', if he is inside b', or at a vertex x' immediately before or after b'. The following lemma essentially states that if at some point of time the person and dog are both inside a vector gadget, they are forced to traverse both vector gadgets simultaneously at 'the same speed'.

▶ Lemma 2. Let $b \in \mathcal{B}, b' \in \mathcal{B}'$ be boxes such that both $\text{next}(b), \text{next}(b')$ are defined. If currently the person is adjacent to b and the dog is adjacent to b', and they traverse their paths while maintaining a distance of at most 5, then after finitely many steps we have that simultaneously the person is adjacent to $\text{next}(b)$ and the dog is adjacent to $\text{next}(b')$. Furthermore, if $\text{next}(b), \text{next}(b')$ are not locally orthogonal, then it is impossible for the person and dog to continue on their paths.

4 Full construction

We describe the complete reduction from the OV problem to the discrete Fréchet distance. Given an instance of OV with sets $U, V \subseteq \{0,1\}^d$, we first pre-process it for technical reasons. For this, if $V = \{v^{(1)}, \dots, v^{(n)}\}$, consider the binary string $\text{str}(V) := v^{(1)}v^{(2)} \dots v^{(n)} \in \{0,1\}^{nd}$ created by writing down all vectors in V one after another. A *consecutive substring* of some string $s_1 s_2 \dots s_k$ is a substring $s_a s_{a+1} \dots s_{b-1} s_b$ for some $1 \le a < b \le k$.

▶ Lemma 3. An instance $U, V \subseteq \mathbb{R}^d$ of OV can be preprocessed in linear time $O(d(n+m))$, resulting in a new instance $U', V' \subseteq \mathbb{R}^{d'}$ such that

- a yes-instance stays a yes-instance and a no-instance stays a no-instance,
- for all $u' \in U'$, $u'_1 = u'_d = 0$ and for all $v' \in V'$, $v'_1 = v'_d = 1$,
- if the instance is a no-instance, then for all $u' \in U'$ the vector u' is not only non-orthogonal to every $v' \in V'$, but even non-orthogonal to all consecutive substrings of $\text{str}(V')$ of length d.

Let us from now on assume that $U, V \subseteq \mathbb{R}^d$ satisfy the guarantees of the preprocessing. We define a graph G and paths P, Q based on U, V as in Figure 3. For each vector $u \in U$ the graph contains a unique orange vector gadget G_u. For each vector $v \in V$ the graph contains a unique blue vector gadget G_v. Before every orange vector gadget there is a vertex A, and after every orange vector gadget there are vertices y_1, y_2, B. Consecutive blue gadgets are simply connected by connecting the last and first $1'$ of the respective gadgets. Finally, vertices $\alpha, \alpha^\star, z', \beta, \beta^\star$, and paths between α and $\alpha^\star$ as well as between β and $\beta^\star$ are added to the graph. This completes the description of the planar graph G. The orange path P starts at A before the first orange gadget and ends at B after the last orange gadget and traverses the orange gadgets in-between. The blue path Q goes from α to $\alpha^\star$, traverses the blue gadgets, then goes from z' to $\beta^\star$ to β. This completes the description of P and Q.

Observe the following properties: Vertices A or B have all vertices of all blue vector gadgets entirely within distance 5. Vertices α and β have all vertices of all orange vector gadgets entirely within distance 5. On the other hand, if the dog is standing on $\alpha^\star$, the only vertices of P within a reach of 5 are vertices labelled A. Similarly, if the dog stands on $\beta^\star$, the only vertices of P within a reach of 5 are vertices labelled B.

▶ Lemma 4. If (U, V) is a yes-instance of OV, then $\text{D}_{\mathcal{F}}(P, Q) \le 5$.

Proof sketch. If $u \in U$ is orthogonal to $v \in V$, the person and the dog can employ the following strategy: The dog stays at α, while the person goes to the orange gadget G_u corresponding to u and waits at A before gadget G_u. Then the dog walks over $\alpha^\star$ to the blue gadget corresponding to v. Then, the person and the dog simultaneously move to the first box of their respective gadget, and traverse their gadget in unison (which is possible due to Lemma 1 since u, v are orthogonal). The dog waits while the person goes via y_1 and

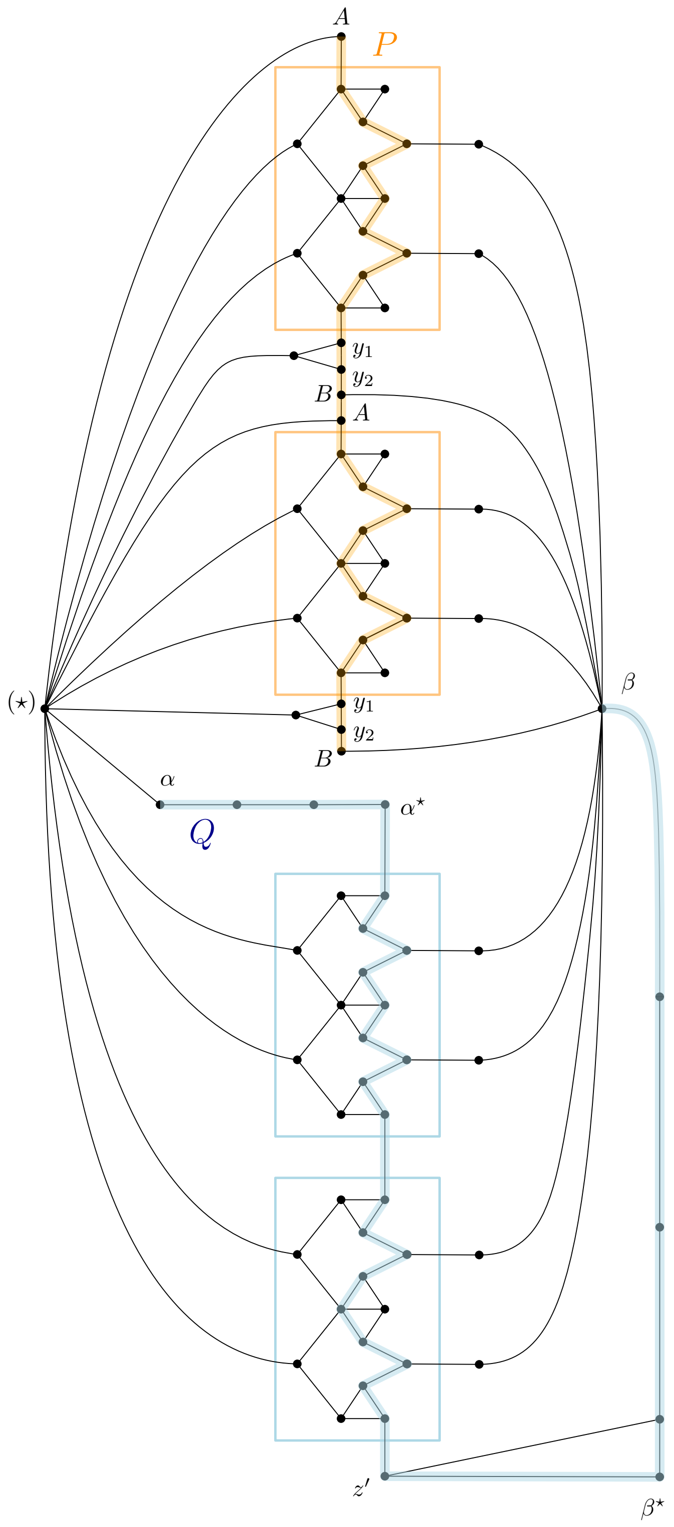


Figure 3 Complete construction of the reduction from OVH, for the two sets $U = \{(0,1,0),(0,0,0)\}$ and $V = \{(1,1,1),(1,0,1)\}$.

y_2 to the vertex B immediately after G_u. Then the person waits while the dog walks all of the remaining path Q to β. Finally, the person walks all of the remaining path P. It can be checked that this maintains a maximum distance of 5. ◀

▶ Lemma 5. If $\mathrm{D}_{\mathcal{F}}(P, Q) \leq 5$, then (U, V) is a yes-instance of OV.

Proof sketch. Consider a traversal of P and Q where at all times, the person and the dog are within distance 5 of one another. At some point of time, the dog is on $\alpha^\star$. This implies the person is on a vertex A right before some orange gadget G_u corresponding to some $u \in U$. Now the person cannot move on his own, so the dog has to move. The dog may walk through the blue gadgets for some time on his own, but at some point of time t_0 the person has also to move away from A into G_u to either 0 or 1. Since z' is at distance greater than 5 from 0 and 1, this time t_0 occurs before the dog finished traversing all the blue gadgets. But now due to Lemma 2 the person and the dog have to move *with the same speed* at least until the person has passed G_u. This implies that u is orthogonal to a consecutive substring of length d of the string $\mathrm{str}(V)$. Due to the preprocessing (Lemma 3), this actually means that (U, V) is a yes-instance. ◀

References

1 Pankaj K Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the discrete fréchet distance in subquadratic time. *SIAM Journal on Computing*, 43(2):429–449, 2014.

2 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(01n02):75–91, 1995.

3 Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On map-matching vehicle tracking data. In *Proceedings of the 31st international conference on Very large data bases*, pages 853–864, 2005.

4 Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 661–670. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.76.

5 Karl Bringmann and Wolfgang Mulzer. Approximability of the discrete Fréchet distance. *Journal of Computational Geometry*, 7(2):46–76, 2016.

6 Kevin Buchin, Maike Buchin, David Duran, Brittany Terese Fasy, Roel Jacobs, Vera Sacristan, Rodrigo I Silveira, Frank Staals, and Carola Wenk. Clustering trajectories for map construction. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 1–10, 2017.

7 Kevin Buchin, Tim Ophelders, and Bettina Speckmann. Seth says: Weak Fréchet distance is faster, but only if it is continuous and in one dimension. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2887–2901. SIAM, 2019.

8 Maike Buchin, Bernhard Kilgus, and Andrea Kölzsch. Group diagrams for representing trajectories. *International Journal of Geographical Information Science*, 34(12):2401–2433, 2020.

9 Thomas Devogele. A new merging process for data integration based on the discrete Fréchet distance. In *Advances in spatial data handling*, pages 167–181. Springer, 2002.

10 Anne Driemel, Ivor van der Hoog, and Eva Rotenberg. On the discrete fréchet distance in a graph. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry, SoCG 2022, June 7-10, 2022, Berlin, Germany*, volume 224 of *LIPIcs*, pages 36:1–36:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. URL: `https://doi.org/10.4230/LIPIcs.SoCG.2022.36`, `doi:10.4230/LIPICS.SOCG.2022.36`.

11 Maximilian Konzack, Thomas McKetterick, Tim Ophelders, Maike Buchin, Luca Giuggioli, Jed Long, Trisalyn Nelson, Michel A Westenberg, and Kevin Buchin. Visual analytics of delays and interaction in movement data. *International Journal of Geographical Information Science*, 31(2):320–345, 2017.

12 Ariane Mascret, Thomas Devogele, Iwan Le Berre, and Alain Hénaff. Coastline matching process based on the discrete Fréchet distance. In *Progress in Spatial Data Handling*, pages 383–400. Springer, 2006.

13 Roniel S. De Sousa, Azzedine Boukerche, and Antonio A. F. Loureiro. Vehicle trajectory similarity: Models, methods, and applications. *ACM Comput. Surv.*, 53(5), September 2020. `doi:10.1145/3406096`.

14 E Sriraghavendra, K Karthik, and Chiranjib Bhattacharyya. Fréchet distance based approach for searching online handwritten documents. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 1, pages 461–465. IEEE, 2007.

15 Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. A survey of trajectory distance measures and performance evaluation. *The VLDB Journal*, 29(1):3–32, 2020.

16 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. URL: `https://doi.org/10.1016/j.tcs.2005.09.023`, `doi:10.1016/J.TCS.2005.09.023`.

17 Dong Xie, Feifei Li, and Jeff M Phillips. Distributed trajectory similarity search. *Proceedings of the VLDB Endowment*, 10(11):1478–1489, 2017.

On the Twin-Width of Smooth Manifolds*

Édouard Bonnet[1] and Kristóf Huszár[2]

1 **Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France**
`edouard.bonnet@ens-lyon.fr`

2 **Institute of Geometry, Graz University of Technology, Austria**
`kristof.huszar@tugraz.at`

Abstract

Building on Whitney's classical method of triangulating smooth manifolds, we show that every compact d-dimensional smooth manifold admits a triangulation with dual graph of twin-width at most $d^{O(d)}$. In particular, it follows that every compact 3-manifold has a triangulation with dual graph of bounded twin-width. This is in sharp contrast to the case of treewidth, where for any natural number n there exists a closed 3-manifold such that every triangulation thereof has dual graph with treewidth at least n. To establish this result, we bound the twin-width of the incidence graph of the d-skeleton of the second barycentric subdivision of the $2d$-dimensional hypercubic honeycomb. We also show that every compact, piecewise-linear (hence smooth) d-dimensional manifold has triangulations where the dual graph has an arbitrarily large twin-width.

Related Version *Full Version:* `https://arxiv.org/abs/2407.10174` [6]

1 Introduction

Structural graph parameters have become increasingly important in computational topology in the past two decades. This is mainly due to the emergence of fixed-parameter tractable (FPT) algorithms for problems on knots, links [9, 28, 29, 31] and 3-manifolds [11, 13, 14, 15, 16], most of which are known to be NP-hard in general. Although these FPT algorithms may have exponential worst-case running time, on inputs with bounded *treewidth* they are guaranteed to terminate in polynomial time. In addition, some of these algorithms have been implemented in software packages such as `Regina`, providing practical tools for topologists [8, 10].

The success of the above algorithms naturally leads to the following question. Given a 3-manifold $\mathcal{M}$ (resp. knot $\mathcal{K}$), what is the smallest treewidth that the dual graph of a triangulation of $\mathcal{M}$ (resp. a diagram of $\mathcal{K}$) may have?[1] Motivated by this challenge, in recent years several results have been obtained that reveal quantitative connections between *topological invariants* of knots and 3-manifolds, and *width parameters* associated with their diagrams [18, 27] and triangulations [21, 22, 23, 24, 25, 32], respectively. While topological properties of 3-manifolds may prohibit the existence of "thin" triangulations [24, 25], geometric or topological descriptions of 3-manifolds can also give strong hints on how to triangulate them so that their dual graphs have constant pathwidth or treewidth, or at least bounded in terms of a topological invariant of these 3-manifolds [20, 22, 23, 32].

In this work we establish similar results for another graph parameter called *twin-width*. Introduced in [7], this notion has been subject of growing interest and found many algorithmic applications in recent years [3]. Namely, on classes of effectively bounded twin-width, first-order properties can be decided efficiently [7], first-order queries can be enumerated fast [19],

* The authors have been supported by the French National Research Agency through the grant ANR-21-CE48-0014. K.H. was partially supported by the Austrian Science Fund (FWF) grant P 33765-N.

[1] For links and knots, this question was respectively asked in [29, Section 4] and [12, p. 2694].

and enhanced approximation algorithms can be designed for several graph optimization problems [1]. Besides, classes of bounded twin-width are fairly general and broad. They for instance include classes of bounded tree-width, and even its dense analogue, clique-width, classes excluding a minor, proper permutation classes, d-dimensional grid graphs [7], some classes of cubic expanders [5], segment intersection graphs without biclique subgraphs of a fixed size [4], and so-called first-order transductions of all these classes [7].

Our first result shows that a compact d-dimensional smooth manifold always has a triangulation with dual graph of twin-width bounded in terms of d.

▶ **Theorem 1.** *Any compact d-dimensional smooth manifold $\mathcal{M}$ admits a triangulation (more precisely, a* Whitney triangulation*, cf. Section 3) with dual graph of twin-width at most $d^{O(d)}$.*

Let us emphasize that, in an appropriate computational model, Whitney triangulations of smooth manifolds can be constructed algorithmically [2]. As every 3-manifold is smooth [33] (see also [30]), the next corollary follows immediately from Theorem 1.

▶ **Corollary 2.** *There exists a universal constant $C > 0$ such that every compact 3-dimensional manifold $\mathcal{M}$ admits a triangulation $\mathcal{T}$ with dual graph $\Gamma(\mathcal{T})$ of twin-width $\operatorname{tww}(\Gamma(\mathcal{T})) \leqslant C$.*

This is in sharp contrast to the case of treewidth, for which it is known that for every $n \in \mathbb{N}$ there are infinitely many 3-manifolds where the smallest treewidth of the dual graph of *every* triangulation is at least n [24, 25]. Complementing Theorem 1, in the full version we also show that for any fixed $d \geqslant 3$, the d-dimensional triangulations of large twin-width are abundant [6, Theorem 19], moreover, any piecewise-linear (hence smooth [17, 35]) manifold of dimension at least three admits triangulations with dual graph of arbitrarily large twin-width:

▶ **Theorem 3.** *Let $d \geqslant 3$ be an integer. For every compact d-dimensional piecewise-linear manifold $\mathcal{M}$ and natural number $n \in \mathbb{N}$, there is a triangulation $\mathcal{T}$ of $\mathcal{M}$ with $\operatorname{tww}(\Gamma(\mathcal{T})) \geqslant n$.*

The proof of Theorem 3 has two stages: First, we show that the d-dimensional ball has triangulations with a dual graph of arbitrarily large twin-width [6, Theorem 21]. Second, we extend the result to every piecewise-linear (PL) manifold. For this we rely on the monotonicity of twin-width with respect to taking induced subtrigraphs (Proposition 4). We establish [6, Theorem 21] using the fact that the class of d-subdivisions of $(d+1)$-regular graphs is not *small* [6, Proposition 7] together with classical results from the theory of PL-manifolds. For details of the proof of Theorem 3 we refer to Section 5.2 of the full version [6].

2 Preliminaries

2.1 Trigraphs, contraction sequences and twin-width

Following [7, Sections 3 and 4], below we review the graph-theoretic notions we rely on.

Trigraphs. A *trigraph* G is a triple $G = (V, E, R)$, where $V = V(G)$ is a finite set of *vertices*, and $E, R \subseteq \binom{V}{2}$ are two disjoint subsets of pairs of vertices called *black edges* and *red edges*, respectively. For a vertex $v \in V(G)$ the *degree* $\deg(v)$ *of* v is the number of edges incident to it, i.e., $\deg(v) = |\{e \in E \cup R : v \in e\}|$. Additionally, the *red degree* $\deg_R(v)$ *of* v is the number of red edges incident to it, i.e., $\deg_R(v) = |\{e \in R : v \in e\}|$. A trigraph G for which $\deg_R(G) = \max_{v \in V(G)} \deg_R(v) \leqslant b$ is called a *b-trigraph*.

Contraction sequences and twin-width. Let $G = (V, E, R)$ be a trigraph and $u, v \in V$ be two arbitrary distinct vertices of G. We say that the trigraph $G/u, v = (V', E', R')$ is obtained from G by *contracting* u and v into a new vertex w if **1.** $V' = (V \setminus \{u, v\}) \cup \{w\}$, **2.** $G - \{u, v\} = (G/u, v) - \{w\}$ and **3.** for any $x \in V' \setminus \{w\} = V \setminus \{u, v\}$ we have

- $\{w, x\} \in E'$ if and only if $\{u, x\} \in E$ and $\{v, x\} \in E$,
- $\{w, x\} \notin E' \cup R'$ if and only if $\{u, x\} \notin E \cup R$ and $\{v, x\} \notin E \cup R$, and
- $\{w, x\} \in R'$ otherwise.

We call the trigraph $G/u, v$ a *contraction* of G. A sequence $\mathfrak{S} = (G_1, \dots, G_m)$ of trigraphs is a *contraction sequence* if G_{i+1} is a contraction of G_i for every $1 \leqslant i \leqslant m - 1$. Note that $|V(G_{i+1})| = |V(G_i)| - 1$. The *width* $\mathrm{w}(\mathfrak{S})$ of a contraction sequence $\mathfrak{S} = (G_1, \dots, G_m)$ is defined as $\mathrm{w}(\mathfrak{S}) = \max_{1 \leqslant i \leqslant m} \deg_R(G_i)$, i.e., the largest red degree of any vertex of any trigraph in $\mathfrak{S}$. Now the *twin-width* $\mathrm{tww}(G)$ of a trigraph G is the smallest width of any contraction sequence $(G_1, \dots, G_{|V(G)|})$, where $G_1 = G$ and $G_{|V(G)|}$ consists of a single vertex.

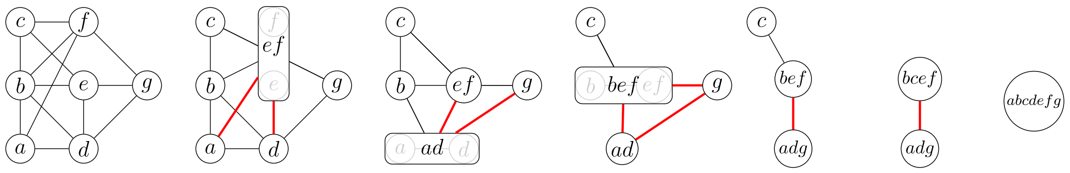


Figure 1 Left to right: a contraction sequence of width two. Reproduced from [3, p. 24].

The next proposition is a simple consequence of the definitions (cf. [7, Section 4.1]).

▶ **Proposition 4.** *If H is an induced subtrigraph of a trigraph G, then* $\mathrm{tww}(H) \leqslant \mathrm{tww}(G)$.

The *d-dimensional n-grid with diagonals* $D_{n,d}$ is the graph with $V(D_{n,d}) = [n]^d$, and $\{u, v\} \in E(D_{n,d})$ for two vertices $u = (u_1, \dots, u_d)$ and $v = (v_1, \dots, v_d)$ if and only if $\max_{i=1}^{d} |u_i - v_i| \leqslant 1$. Now, for a given trigraph $G = (V, E, R)$ we set $\mathrm{red}(G) = (V, \emptyset, E \cup R)$. In words, $\mathrm{red}(G)$ is the trigraph obtained from G by replacing every black edge of G by a red edge between the same vertices. With this notation $\mathrm{red}(D_{n,d})$ is the *d-dimensional* ***red*** *n-grid with diagonals*, i.e., $\mathrm{red}(D_{n,d}) = ([n]^d, \emptyset, E(D_{n,d}))$. Clearly, $\mathrm{tww}(D_{n,d}) \leqslant \mathrm{tww}(\mathrm{red}(D_{n,d}))$.

▶ **Theorem 5** (Lemma 4.4 in [7]). *For every positive d and n, every subtrigraph of the d-dimensional red n-grid with diagonals* $\mathrm{red}(D_{n,d})$ *has twin-width at most* $2(3^d - 1)$.

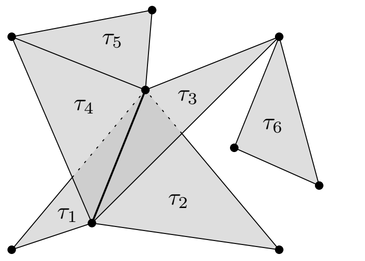
2.2 Some background in topology

Simplicial and cubical complexes. Given a finite ground set $\mathcal{S}$, a *simplicial complex* $\mathcal{X}$ over $\mathcal{S}$ is subset of the power set $2^{\mathcal{S}}$, such that $\mathcal{F} \in \mathcal{X}$ and $\mathcal{F}' \subset \mathcal{F}$ imply $\mathcal{F}' \in \mathcal{X}$. Any element of $\mathcal{X}$ is called a *face* or *simplex* of $\mathcal{X}$, and for $\sigma \in \mathcal{X}$ the *dimension of* σ is defined as $\dim \sigma = |\sigma| - 1$. The *dimension of* $\mathcal{X}$ is defined as $\dim \mathcal{X} = \max_{\sigma \in \mathcal{X}} \dim \sigma$. If $\dim \mathcal{X} = d$, we also say that $\mathcal{X}$ is a *simplicial d-complex*. For $0 \leqslant i \leqslant \dim \mathcal{X}$, we let $\mathcal{X}(i) = \{\sigma \in \mathcal{X} : \dim \sigma = i\}$. The *$i$-skeleton* $\mathcal{X}_i = \bigcup_{j=0}^{i} \mathcal{X}(j)$ of $\mathcal{X}$ is the union of all faces of $\mathcal{X}$ up to dimension i.

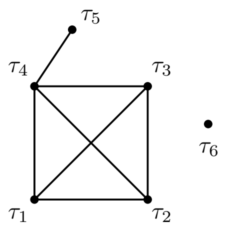
Analogous to simplicial complexes, a *cubical complex* $\mathcal{X}$ over a ground set $\mathcal{S}$ is a set system $\mathcal{X} \subset 2^{\mathcal{S}}$ that consists of "cubes" instead of simplices. The terminology is the same as in the simplicial case. Cubical or simplicial complexes in this work will typically be defined geometrically, and as such they will naturally come with a *geometric realization.*

We let $\mathcal{X}'$ denote the *barycentric subdivision* of a cubical or simplicial complex $\mathcal{X}$. Note that $\mathcal{X}'$ is always a simplicial complex. For further background we refer to [34].

Pure complexes and their dual graphs. A complex $\mathcal{X}$ is *pure* if every face of $\mathcal{X}$ is contained in a face of dimension $\dim \mathcal{X}$. It follows that, for every i with $0 \leqslant i \leqslant \dim \mathcal{X}$, the i-skeleton $\mathcal{X}_i$ of a pure complex $\mathcal{X}$ is also pure. Given a pure complex $\mathcal{X}$, the *dual graph* $\Gamma(\mathcal{X}_i) = (V, E)$ of its i-skeleton is defined as the graph, where the vertex set V corresponds to the set $\mathcal{X}(i)$ of i-faces, and $\{\sigma, \tau\} \in E$ if and only if σ and τ share an $(i-1)$-dimensional face in $\mathcal{X}$.



(a) A pure simplicial 2-complex $\mathcal{X}$ formed by six triangles τ_1, τ_2, τ_3, τ_4, τ_5, and τ_6.



(b) The dual graph $\Gamma(\mathcal{X}_2)$ of $\mathcal{X}_2$.

Figure 2 Example of a 2-dimensional pure simplicial complex and its dual graph.

The hypercubic honeycomb. Let n and d be positive integers and consider the d-dimensional cube $[1,n]^d \subset \mathbb{R}^d$. The *$d$-dimensional hypercubic honeycomb* $\mathbf{H}^{d,n}$ is a pure cubical complex that decomposes $[1,n]^d$ into $(n-1)^d$ geometric cubes, see Figure 3. The properties of this familiar object (cf. Theorem 9) play an important role in establishing our main result.

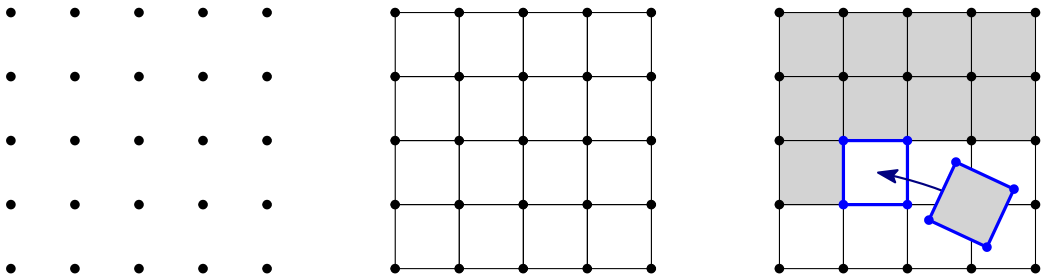


Figure 3 Constructing the complex $\mathbf{H}^{2,5}$. Its 1-skeleton $\mathbf{H}_1^{2,5}$ is isomorphic to the (5×5)-grid.

3 Whitney's method

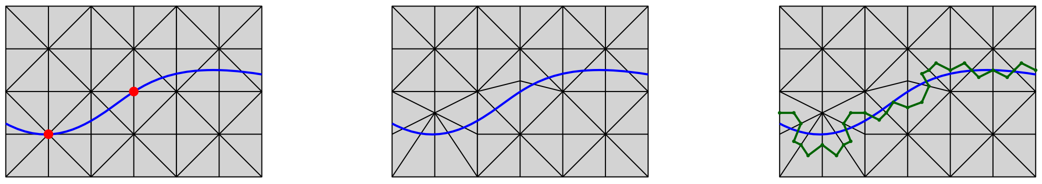
Two seminal results of Whitney state that a smooth d-dimensional manifold $\mathcal{M}$ always admits a smooth embedding in $\mathbb{R}^{2d}$, which can then be used to obtain a triangulation of $\mathcal{M}$.

▶ **Theorem 6** (strong Whitney embedding theorem [36, Theorem 5], cf. [26, Theorem 6.19])**.** *For $d > 0$, every smooth d-manifold admits a smooth embedding into $\mathbb{R}^{2d}$.*

▶ **Theorem 7** (triangulation theorem [37, Chapter IV.B], cf. [2, Theorem 1.1])**.** *Every compact, smooth d-manifold $\mathcal{M}$ embedded in some Euclidean space $\mathbb{R}^m$ admits a triangulation.*

Next, we give a high-level overview of Whitney's method of triangulating embedded smooth manifolds based on [37, Chapter IV.B], which is sufficient for our purposes (cf. [2]).

Triangulating smooth manifolds. Let $\mathcal{M}$ be a compact smooth d-manifold. By Theorem 6 there exists a smooth embedding $\iota\colon \mathcal{M} \to \mathbb{R}^{2d}$. Given such an embedding, we choose a sufficiently fine (with respect to ι) hypercubic honeycomb decomposition of $\mathbb{R}^{2d}$, denoted by L_0. Next, we pass to the first barycentric subdivision L of L_0. (Geometrically, L is obtained from L_0 by subdividing each k-dimensional hypercube of L_0 into $(2k)!!$ simplices.) By slightly perturbing the vertices of L we obtain a triangulation L^* of $\mathbb{R}^{2d}$, which is combinatorially isomorphic to L, but is in *general position* with respect to $\iota(\mathcal{M}) \subset \mathbb{R}^{2d}$. Now L^* induces a triangulation $\mathcal{T}$ of $\mathcal{M}$, where, importantly, $\mathcal{T}$ is a subcomplex of the d-skeleton of the barycentric subdivision $(L^*)'$ of L^*. See Figure 4 for an example.



(a) The triangulation L and the image $\iota(\mathcal{M})$ (blue). The red points indicate the vertices of L contained by $\iota(\mathcal{M})$.

(b) The perturbed triangulation L^*, which in general position with respect to $\iota(\mathcal{M})$.

(c) The resulting triangulation $\mathcal{T}$ of $\mathcal{M}$ (dark green). $\mathcal{T}$ is a subcomplex of $(L^*)'$.

Figure 4 Illustration of Whitney's method of triangulating ambient submanifolds ($d = 1$).

4 Outline of the proof of Theorem 1

Here we outline the proof of Theorem 1, i.e., every compact smooth d-manifold has a triangulation $\mathcal{T}$ with $\operatorname{tww}(\Gamma(\mathcal{T})) \leqslant d^{O(d)}$. To simplify the notation, we let $G_{d,n} = \Gamma(((\mathbf{H}^{2d,n})'')_d)$, i.e., $G_{d,n}$ denotes the dual graph of the d-skeleton of the second barycentric subdivision of the hybercubic honeycomb $\mathbf{H}^{2d,n}$. The result is based on the following properties of $G_{d,n}$.

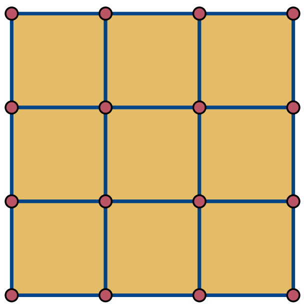
▶ **Proposition 8.** *For any Whitney triangulation $\mathcal{T}$ of a compact smooth d-manifold $\mathcal{M}$, we have that $\mathcal{T}$ is a subcomplex of $((\mathbf{H}^{2d,n})'')_d$. Hence, its dual graph $\Gamma(\mathcal{T})$ is an **induced** subgraph of $G_{d,n} = \Gamma(((\mathbf{H}^{2d,n})'')_d)$.*

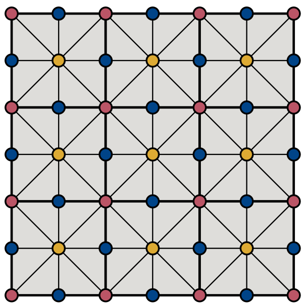
▶ **Theorem 9.** *The twin-width of the dual graph $G_{d,n} = \Gamma(((\mathbf{H}^{2d,n})'')_d)$ of the d-skeleton of the second barycentric subdivision of the hypercubic honeycomb $\mathbf{H}^{2d,n}$ satisfies* $\operatorname{tww}(G_{d,n}) \leqslant d^{O(d)}$.

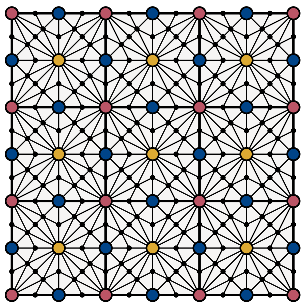
Proposition 8 is a direct consequence of Whitney's construction. Before outlining the proof of Theorem 9, we show how these statements imply Theorem 1.

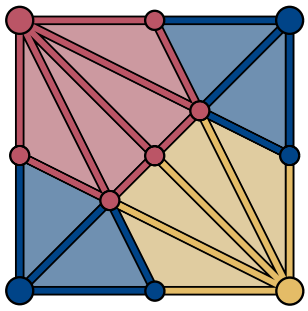
Proof of Theorem 1. Let $\mathcal{M}$ be a compact, smooth d-dimensional manifold. Consider a Whitney triangulation $\mathcal{T}$ of $\mathcal{M}$. By Proposition 8, $\Gamma(\mathcal{T})$ is an induced subgraph of $G_{d,n}$ and by Theorem 9, $\operatorname{tww}(G_{d,n}) \leqslant d^{O(d)}$. Hence, since twin-width is monotone under taking induced subtrigraphs (Proposition 4), we obtain $\operatorname{tww}(\Gamma(\mathcal{T})) \leqslant \operatorname{tww}(G_{d,n}) \leqslant d^{O(d)}$. ◀

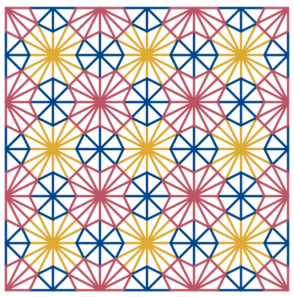
Outline of the proof of Theorem 9. For bookkeeping purposes, we color all faces of $\mathbf{H}^{2d,n}$ by their dimension, see Figure 5(a). This induces a coloring of the simplices of $((\mathbf{H}^{2d,n})'')_d$, as explained in Figures 5(b)–(e). Now we construct a contraction sequence $\mathfrak{S}$ of $G_{d,n}$ by first contracting the vertices of $G_{d,n}$ corresponding to the d-simplices of a connected monochromatic component of $((\mathbf{H}^{2d,n})'')_d$ to a single vertex (in any order). This gives a trigraph $G^*_{d,n}$ (Figure 5(f)), which turns out to be a subtrigraph of $\operatorname{red}(D_{n,2d})$, the $2d$-dimensional red n-grid with diagonals. We complete $\mathfrak{S}$ with an optimal contraction sequence for $G^*_{d,n}$. It can be shown that the first part of $\mathfrak{S}$ (which contracts $G_{d,n}$ to $G^*_{d,n}$) has width bounded by $d^{O(d)}$ (see [6, Section 4]). Furthermore, by Theorem 5 $\operatorname{tww}(G^*_{d,n}) \leqslant 2(3^{2d} - 1)$. But this implies that $\operatorname{w}(\mathfrak{S}) \leqslant d^{O(d)}$. It follows that $\operatorname{tww}(G_{d,n}) \leqslant d^{O(d)}$. ◀


(a) The hypercubic honeycomb $\mathbf{H}^{2d,n}$ with its cells colored by their dimension.


(b) The complex $(\mathbf{H}^{2d,n})'$ with vertices colored as their corresponding cells in $\mathbf{H}^{2d,n}$.


(c) The complex $(\mathbf{H}^{2d,n})''$. Vertices corresponding to those in $(\mathbf{H}^{2d,n})'$ are colored as before.


(d) An extension of the coloring to all simplices of $(\mathbf{H}^{2d,n})''$ (shown near the top left corner). The four larger disks correspond to vertices in $(\mathbf{H}^{2d,n})'$.


(e) The extended coloring restricted to the d-simplices of $(\mathbf{H}^{2d,n})''$ gives a partition of the d-simplices into connected monochromatic components.

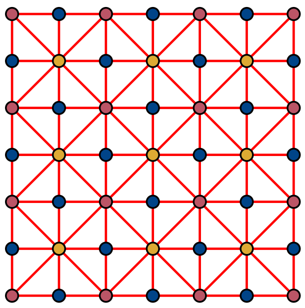

(f) Contracting each monochromatic component to a single node yields a trigraph $G^*_{d,n}$ with all edges red. It is known that $\operatorname{tww}(G^*_{d,n}) \leqslant 2(3^{2d} - 1)$.

Figure 5 **(a)–(c)** Illustrations of the cubical complex $\mathbf{H}^{2d,n}$ for $d = 1$ and $n = 4$, and of its first and second barycentric subdivisions (which are simplicial complexes) displaying the colorings described in the proof of Theorem 9. **(d)–(f)** Three essential steps in the proof of Theorem 9.

References

1 P. Bergé, É. Bonnet, H. Déprés, and R. Watrigant. Approximating highly inapproximable problems on graphs of bounded twin-width. In *40th Int. Symp. Theor. Aspects Comput. Sci. (STACS 2023)*, volume 254 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 10:1–10:15. Schloss Dagstuhl. Leibniz-Zent. Inform., 2023. `doi:10.4230/lipics.stacs.2023.10`.

2 J.-D. Boissonnat, S. Kachanovich, and M. Wintraecken. Triangulating submanifolds: an elementary and quantified version of Whitney's method. *Discrete Comput. Geom.*, 66(1):386–434, 2021. `doi:10.1007/s00454-020-00250-8`.

3 É. Bonnet. *Twin-width and contraction sequences.* Habilitation thesis, ENS de Lyon, April 2024. URL: `https://perso.ens-lyon.fr/edouard.bonnet/text/hdr.pdf`.

4 É. Bonnet, D. Chakraborty, E. J. Kim, N. Köhler, R. Lopes, and S. Thomassé. Twin-width VIII: delineation and win-wins. In *17th Int. Symp. Parametr. Exact Comput. (IPEC*

2022), volume 249 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 9:1–9:18. Schloss Dagstuhl. Leibniz-Zent. Inform., 2022. doi:10.4230/lipics.ipec.2022.9.

5 É. Bonnet, C. Geniet, E. J. Kim, S. Thomassé, and R. Watrigant. Twin-width II: small classes. *Comb. Theory*, 2(2):Paper No. 10, 42, 2022. doi:10.5070/C62257876.

6 É. Bonnet and K. Huszár. On the twin-width of smooth manifolds, 2024. arXiv:2407.10174.

7 É. Bonnet, E. J. Kim, S. Thomassé, and R. Watrigant. Twin-width I: Tractable FO model checking. *J. ACM*, 69(1):Art. 3, 46, 2022. doi:10.1145/3486655.

8 B. A. Burton. Computational topology with Regina: algorithms, heuristics and implementations. In *Geometry and Topology Down Under*, volume 597 of *Contemp. Math.*, pages 195–224. Am. Math. Soc., Providence, RI, 2013. doi:10.1090/conm/597/11877.

9 B. A. Burton. The HOMFLY-PT polynomial is fixed-parameter tractable. In *34th Int. Symp. Comput. Geom. (SoCG 2018)*, volume 99 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 18:1–18:14. Schloss Dagstuhl–Leibniz-Zent. Inf., 2018. doi:10.4230/LIPIcs.SoCG.2018.18.

10 B. A. Burton, R. Budney, W. Pettersson, et al. Regina: Software for low-dimensional topology, 1999–2023. Version 7.3. URL: https://regina-normal.github.io.

11 B. A. Burton and R. G. Downey. Courcelle's theorem for triangulations. *J. Comb. Theory, Ser. A*, 146:264–294, 2017. doi:10.1016/j.jcta.2016.10.001.

12 B. A. Burton, H. Edelsbrunner, J. Erickson, and S. Tillmann, editors. *Computational Geometric and Algebraic Topology*, volume 12 of *Oberwolfach Rep.* EMS Publ. House, 2015. doi:10.4171/OWR/2015/45.

13 B. A. Burton, T. Lewiner, J. Paixão, and J. Spreer. Parameterized complexity of discrete Morse theory. *ACM Trans. Math. Softw.*, 42(1):6:1–6:24, 2016. doi:10.1145/2738034.

14 B. A. Burton, C. Maria, and J. Spreer. Algorithms and complexity for Turaev–Viro invariants. *J. Appl. Comput. Topol.*, 2(1–2):33–53, 2018. doi:10.1007/s41468-018-0016-2.

15 B. A. Burton and W. Pettersson. Fixed parameter tractable algorithms in combinatorial topology. In *Proc. 20th Int. Conf. Comput. Comb. (COCOON 2014)*, volume 8591 of *Lect. Notes Comput. Sci.*, pages 300–311. Springer, 2014. doi:10.1007/978-3-319-08783-2_26.

16 B. A. Burton and J. Spreer. The complexity of detecting taut angle structures on triangulations. In *Proc. 24th Annu. ACM-SIAM Symp. Discrete Algorithms (SODA 2013)*, pages 168–183, 2013. doi:10.1137/1.9781611973105.13.

17 S. S. Cairns. On the triangulation of regular loci. *Ann. of Math. (2)*, 35(3):579–587, 1934. doi:10.2307/1968752.

18 A. de Mesmay, J. Purcell, S. Schleimer, and E. Sedgwick. On the tree-width of knot diagrams. *J. Comput. Geom.*, 10(1):164–180, 2019. doi:10.20382/jocg.v10i1a6.

19 J. Gajarský, M. Pilipczuk, W. Przybyszewski, and Sz. Toruńczyk. Twin-width and types. In *49th Int. Conf. Autom. Lang. Prog. (ICALP 2022)*, volume 229 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 123:1–123:21. Schloss Dagstuhl. Leibniz-Zent. Inform., 2022. doi:10.4230/lipics.icalp.2022.123.

20 A. He, J. Morgan, and E. K. Thompson. An algorithm to construct one-vertex triangulations of Heegaard splittings, 2023. arXiv:2312.17556.

21 K. Huszár. *Combinatorial width parameters for 3-dimensonal manifolds.* PhD thesis, IST Austria, June 2020. doi:10.15479/AT:ISTA:8032.

22 K. Huszár. On the pathwidth of hyperbolic 3-manifolds. *Comput. Geom. Topol.*, 1(1):1–19, 2022. doi:10.57717/cgt.v1i1.4.

23 K. Huszár and J. Spreer. 3-Manifold triangulations with small treewidth. In *35th Int. Symp. Comput. Geom. (SoCG 2019)*, volume 129 of *LIPIcs. Leibniz Int. Proc. Inf.*, pages 44:1–44:20. Schloss Dagstuhl–Leibniz-Zent. Inf., 2019. doi:10.4230/LIPIcs.SoCG.2019.44.

24 K. Huszár and J. Spreer. On the width of complicated JSJ decompositions. In *39th Int. Symp. Comput. Geom. (SoCG 2023)*, volume 258 of *LIPIcs. Leibniz Int. Proc. Inf.*, pages 42:1–42:18. Schloss Dagstuhl–Leibniz-Zent. Inf., 2023. doi:10.4230/LIPIcs.SoCG.2023.42.

25 K. Huszár, J. Spreer, and U. Wagner. On the treewidth of triangulated 3-manifolds. *J. Comput. Geom.*, 10(2):70–98, 2019. `doi:10.20382/jogc.v10i2a5`.

26 J. M. Lee. *Introduction to smooth manifolds*, volume 218 of *Grad. Texts Math.* Springer, New York, second edition, 2013. `doi:10.1007/978-1-4419-9982-5`.

27 C. Lunel and A. de Mesmay. A Structural Approach to Tree Decompositions of Knots and Spatial Graphs. In *39th Int. Symp. Comput. Geom. (SoCG 2023)*, volume 258 of *LIPIcs. Leibniz Int. Proc. Inf.*, pages 50:1–50:16. Schloss Dagstuhl–Leibniz-Zent. Inf., 2023. `doi:10.4230/LIPIcs.SoCG.2023.50`.

28 J. A. Makowsky. Coloured Tutte polynomials and Kauffman brackets for graphs of bounded tree width. *Discrete Appl. Math.*, 145(2):276–290, 2005. `doi:10.1016/j.dam.2004.01.016`.

29 J. A. Makowsky and J. P. Mariño. The parametrized complexity of knot polynomials. *J. Comput. Syst. Sci.*, 67(4):742–756, 2003. Special issue on parameterized computation and complexity. `doi:10.1016/S0022-0000(03)00080-1`.

30 C. Manolescu. Lectures on the triangulation conjecture. In *Proc. 22nd Gökova Geom.-Topol. Conf. (GGT 2015)*, pages 1–38. Int. Press Boston, 2016. URL: `https://gokovagt.org/proceedings/2015/manolescu.html`.

31 C. Maria. Parameterized complexity of quantum knot invariants. In *37th Int. Symp. Comput. Geom. (SoCG 2021)*, volume 189 of *LIPIcs. Leibniz Int. Proc. Inf.*, pages 53:1–53:15. Schloss Dagstuhl–Leibniz-Zent. Inf., 2021. `doi:10.4230/LIPIcs.SoCG.2021.53`.

32 C. Maria and J. Purcell. Treewidth, crushing and hyperbolic volume. *Algebr. Geom. Topol.*, 19(5):2625–2652, 2019. `doi:10.2140/agt.2019.19.2625`.

33 E. E. Moise. Affine structures in 3-manifolds. V. The triangulation theorem and Hauptvermutung. *Ann. Math. (2)*, 56:96–114, 1952. `doi:10.2307/1969769`.

34 V. V. Prasolov. *Elements of combinatorial and differential topology*, volume 74 of *Grad. Stud. Math.* Am. Math. Soc., Providence, RI, 2006. Translated from the 2004 Russian original by Olga Sipacheva. `doi:10.1090/gsm/074`.

35 J. H. C. Whitehead. On C^1-complexes. *Ann. of Math. (2)*, 41:809–824, 1940. `doi:10.2307/1968861`.

36 H. Whitney. The self-intersections of a smooth n-manifold in $2n$-space. *Ann. of Math. (2)*, 45:220–246, 1944. `doi:10.2307/1969265`.

37 H. Whitney. *Geometric Integration Theory.* Princeton Univ. Press, Princeton, NJ, 1957. `doi:10.1515/9781400877577`.

An Approximation Lower Bound for Topologically Stable Stateless Kinetic Euclidean MSTs

Wouter Meulemans[1], **Kevin Verbeek**[1], **and Jules Wulms**[1]

1 **Dept. of Mathematics and Computer Science, TU Eindhoven, The Netherlands**
`[w.meulemans | k.a.b.verbeek | j.j.h.m.wulms]@tue.ml`

Related Version arXiv:1704.08000

1 Introduction

Euclidean minimum spanning trees (EMSTs) are a well-known structure in computational geometry: for a set P of n points in $\mathbb{R}^2$, an EMST T connects the points with (straight-line) edges, whose length corresponds to the Euclidean distance between the connected points. To obtain a minimum spanning tree, the total edge length of T must be minimized.

In this abstract we study the *kinetic* EMST problem, in which the input changes continuously over time: The input set P consists of a set of trajectories: $p_i(t)$ describes the location of point i at time t, for $1 \leq i \leq n$. We are generally interested in efficient data structures to maintain kinetic EMSTs [5], and in bounding the number of combinatorial changes an (optimal) EMST undergoes [1, 4]. In many practical applications of kinetic EMSTs, such as visualization of time-varying data and physical networks, making abrupt changes to the edge set can be confusing, disruptive, or even impossible. In such cases, it is essential that the E(M)ST is *stable*: Small changes in the input, should result in small changes in the output.

Meulemans *et al.* [2] introduced a framework for algorithm stability, with various definitions that address the trade-off between solution quality and stability. Solution quality is measured by the optimization function of the problem in question; in case of EMST the length of a spanning tree is measured. In particular, the *topological stability* of algorithmic problems can be analyzed: an algorithm is topologically stable if its output behaves continuously (albeit possibly at an arbitrary speed) as the input changes over time. For this type of stability analysis, we want to consider (continuous) paths in the *input space* $\mathcal{I}$ of an algorithm $\mathcal{A}$, and see what paths they map to in the *solution space* $\mathcal{S}$ of $\mathcal{A}$. For EMST, $\mathcal{I}$ consists of all possible placements of P in $\mathbb{R}^2$, and $\mathcal{S}$ considers all (combinatorial) spanning trees on P.

To ensure that such continuous paths are well-defined, the framework requires specifying a topology for both $\mathcal{I}$ and $\mathcal{S}$ [3]. For kinetic EMSTs, the natural choice for the topology $\mathcal{T}_\mathcal{I}$

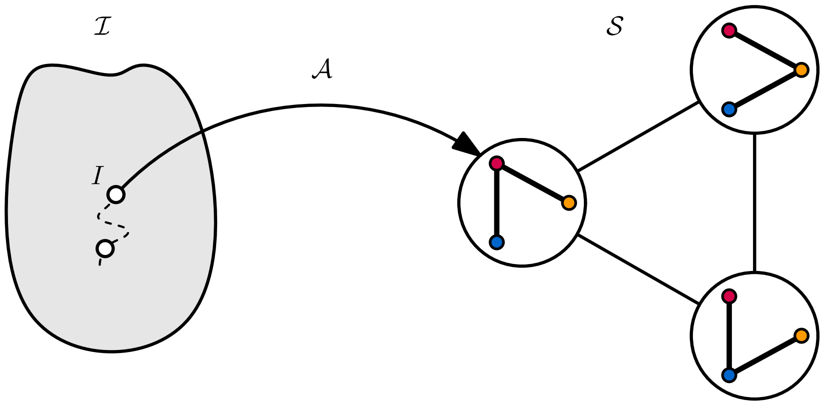


Figure 1 An algorithm $\mathcal{A}$ maps an instance $I \in \mathcal{I}$ to a spanning tree in the flip graph describing $\mathcal{S}$.

of $\mathcal{I}$ is the standard topology on $\mathbb{R}^{2n}$. However, the solution space $\mathcal{S}$ is discrete: spanning trees change by updating the edge set that defines the solution. To facilitate the stability analysis, we need a notion of continuity, which is achieved by defining the topology $\mathcal{T}_\mathcal{S}$ of $\mathcal{S}$ using a *flip graph*: vertices represent (combinatorial descriptions of) spanning trees on P, and edges connect vertices if we consider changing from one spanning tree to the other a continuous change (see Figure 1). We see $\mathcal{S}$ as a simplicial 1-complex of the considered flip graph, allowing paths in $\mathcal{I}$ to be mapped to paths in $\mathcal{S}$. For more details, see [3].

The topological stability ratio ρ_{TS} of a problem is defined as the ratio between the quality of a topologically stable solution and an optimal (and possibly unstable) solution. In [2], the topological stability of kinetic EMSTs is analyzed for *state-aware* algorithms. Such algorithms can maintain a solution that may be updated as the input continues to change. In contrast, we analyze ρ_{TS} of kinetic EMSTs computed by *stateless* algorithms. A stateless algorithm does not maintain a state, that is, a solution that is updated over time, but instead decides beforehand on a solution for each possible input; essentially defining a (static) function.

For stateless algorithms we can prove a lower bound on ρ_{TS} using the fact that a stateless algorithm is a continuous function. Let $\mathcal{A}\colon \mathcal{I} \to \mathcal{S}$ be a continuous function. With slight abuse of notation, we often use a set, for example $D \subseteq \mathcal{I}$ as input for $\mathcal{A}$ to talk about the set $\mathcal{A}D$ of solutions that $\mathcal{A}$ maps the inputs in D to. Specifically, omitting the brackets ensures that this notation corresponds to function composition, for (parameterized) paths in $\mathcal{I}$. In this abstract, we show that such a function $\mathcal{A}$, that computes a topologically stable (approximation of an) EMST on n moving points, produces at least a $\sqrt{n}$-approximation, if $\mathcal{A}$ is a continuous function. In other words, we prove that ρ_{TS} is at least $\Omega(\sqrt{n})$ and we do so for a topology $\mathcal{T}_\mathcal{S}$ of $\mathcal{S}$ defined by *edge flips*: any (combinatorial) edge may be replaced by any other edge, as long as all input points are still connected after the flip (see Figure 1).

▶ **Theorem 1.1.** *For a stateless algorithm $\mathcal{A}$ solving the kinetic EMST problem on n (moving) input points, if $\mathcal{T}_\mathcal{S}$ is defined by edge flips, then $\rho_{\mathrm{TS}}(EMST, \mathcal{T}_\mathcal{I}, \mathcal{T}_\mathcal{S}) = \Omega(\sqrt{n})$.*

Observe that state-aware algorithms can trivially achieve $\rho_{\mathrm{TS}} = 1$ by following an optimal solution, if $\mathcal{T}_\mathcal{S}$ is defined by edge flips, since edge flips suffice to keep an EMST optimal.

2 A Lower Bound on ρ_{TS} of Stateless Kinetic EMSTs

The remainder of this abstract works towards proving Theorem 1.1; omitted proofs are found in Section 4.3 of [3]. Intuitively, we use topological arguments to show that the continuity of $\mathcal{A}$ and the topology of the input and solution spaces prevent function $\mathcal{A}$ from being injective. We carefully choose the geometry of the input instances we consider, such that any spanning tree chosen by $\mathcal{A}$ for the inputs that violate injectivity results in an $\Omega(\sqrt{n})$-approximation.

More specifically, the proof is structured as follows. We assume that $\mathcal{A}$ is a continuous function and consider a part $D \subseteq \mathcal{I}$ of the input space that is homeomorphic to a disk. We show that the boundary δD of this set D must map to a part of $\mathcal{S}$ that cannot have holes, since $\mathcal{A}$ is continuous. More precisely, the image of δD under $\mathcal{A}$ cannot contain cycles of the flip graph $\mathcal{S}$ and hence δD maps to a tree-like part of $\mathcal{S}$. Thus, different parts of δD map to the exact same vertex of $\mathcal{S}$. For those inputs we show an approximation factor of $\Omega(\sqrt{n})$.

For ease of explanation, we consider inputs on point set $P = \{p_1, p_2, \ldots, p_{2n}\}$ consisting of $2n$ moving points in 1-dimensional Euclidean space. This affects our computations only by constant factors compared to inputs with n points; we still work towards proving Theorem 1.1.

Constructing the inputs. We consider a restricted set of inputs, in which the points in P lie on a line, and form sets $P_1 = \{p_1, \ldots, p_n\}$ and $P_2 = \{p_{n+1}, \ldots, p_{2n}\}$ of n points each.

We are going to consider two permutations Π and Π' of both P_1 and P_2; either point set can be in either permutation. Intuitively, Π and Π' correspond to a row-by-row and column-by-column traversal in a $\sqrt{n} \times \sqrt{n}$ grid; see Figure 2a for a visual interpretation.

To create concrete input instances, we consider combinations $X \bowtie Y$ of permutations X and Y of the subsets P_1 and P_2, respectively: For such an instance $X \bowtie Y$, the points in P_1 are ordered according to X, and consecutive points are located at distance 1 from one another. Similarly, the points in P_2 are ordered according to Y, and consecutive points are also at distance 1. The points of P_1 are placed before the points on P_2 and the (last) point $p_n \in P_1$ is at distance n from the (first) point $p_{n+1} \in P_2$. See Figure 2b.

We are now ready to construct our set $D \subseteq \mathcal{I}$ of inputs; see Figure 3a. First, we define the instances on the boundary δD of D: Consider the four instances $I_a = \Pi \bowtie \Pi$, $I_b = \Pi \bowtie \Pi'$, $I_c = \Pi' \bowtie \Pi'$, and $I_d = \Pi' \bowtie \Pi$. The boundary δD consists of four paths $\delta_{ab}, \delta_{bc}, \delta_{cd}$ and δ_{da} through $\mathcal{I}$. The path δ_{ab} has its endpoints at I_a and I_b, and is otherwise defined by a linear interpolation of the positions of points in P_2. Paths δ_{bc}, δ_{cd} and δ_{da} are defined analogously.

Observe that the boundary δD basically forms a square in $\mathcal{I}$. Any point inside the square defines an instance in which P_1 and P_2 are (linearly) interpolating between Π and Π'.

We now prove two crucial lemmata on the length of a spanning tree on certain instances.

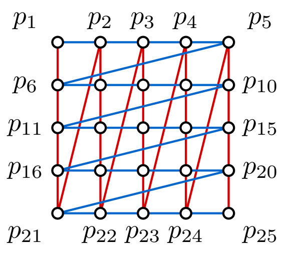
▶ **Lemma 2.1.** *For instances I_a, I_b, I_c and I_d, any spanning tree T that has more than one edge (p, p') with $p \in P_1$ and $p' \in P_2$, can be turned into a spanning tree T' with only one such edge, such that the total edge length of T' is smaller than the total edge length of T.*

Proof sketch. Any additional such edge can be replaced by a shorter edge between two points, either both in P_1 or both in P_2. See Figure 3b. ◀

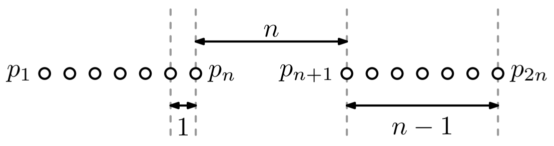
▶ **Lemma 2.2.** *For a set P of n uniformly-spaced collinear points, and an arbitrary (combinatorial) spanning tree T on P, the length of T is an $\Omega(\sqrt{n})$-approximation of the length of an EMST, when the permutation of the points in P is Π or Π'.*

Proof sketch. Consider an arbitrary spanning tree T on n points in permutation Π or Π'.

Via a case distinction we show that an arbitrary edge of T has length $\Omega(\sqrt{n})$ for Π or Π'; if it is shorter for Π, then it has length $\Omega(\sqrt{n})$ for Π', and vice versa. After proving this, it follows that, for Π or Π', at least half of the edges of T are of length $\Omega(\sqrt{n})$, and hence T has a total length of at least $\Omega(n\sqrt{n})$ for that permutation. Observe that an EMST connects the points in the order in which they occur on the line, leading to a length of $\mathrm{OPT} = n - 1$. As such, the length of T is $\Omega(\sqrt{n}) \cdot \mathrm{OPT}$ for one of the two permutations. ◀

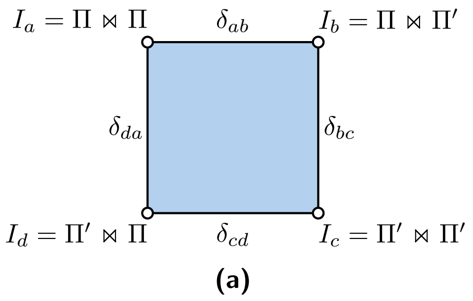


(a)



(b)

Figure 2 Construction in instances for Theorem 1.1: **(a)** The orderings of the two permutations Π and Π' on a small point set with $n = 25$ (in blue and red, respectively). **(b)** A concrete input instance with point sets P_1 and P_2 uniformly spaced, and distance n between the sets.



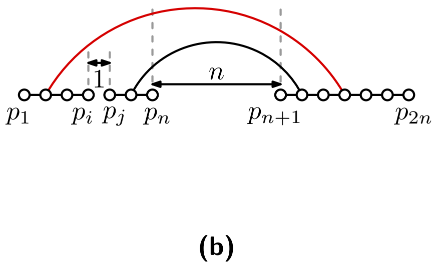


Figure 3 (a) The set of inputs $D \subseteq \mathcal{I}$. **(b)** An instance I_a, I_b, I_c or I_d with multiple edges between P_1 and P_2. These edges are drawn as arcs for clarity, but are straight lines in a geometric spanning tree. Replace the red edge by (p_i, p_j) to shorten the spanning tree.

Mapping to the flip graph. Given our input instances $D \subseteq \mathcal{I}$ we can now investigate the mapping by $\mathcal{A}$ to the flip graph defining $\mathcal{S}$. We start considering specific instances in δD.

▶ **Lemma 2.3.** *Let $I, I' \in \delta D$, with $I \neq I'$, be two instances for which P_1 or P_2 are positioned as follows: the points are collinear, have unit distance between consecutive pairs, and are ordered according to Π in I and according to Π' in I'. If I and I' are mapped by $\mathcal{A}$ such that $\mathcal{A}(I) = \mathcal{A}(I')$, then $\mathcal{A}(I)$ or $\mathcal{A}(I')$ is an $\Omega(\sqrt{n})$-approximation of the EMST.*

Proof sketch. By Lemma 2.1 we may assume that $\mathcal{A}(I)$ and $\mathcal{A}(I')$ are modified to have only a single edge between P_1 and P_2, since this can only shorten their total edge length.

Assume without loss of generality that the conditions in the lemma hold for P_1. Then Lemma 2.2 applies to the subtree on P_1: this is an $\Omega(\sqrt{n})$-approximation for $\mathcal{A}(I)$ or $\mathcal{A}(I')$.

To finish the proof, we show that even if the remainders of $\mathcal{A}(I)$ and $\mathcal{A}(I')$ are as short as possible, one of them (in its entirety) is still a $\Omega(\sqrt{n})$-approximation. The edge between P_1 and P_2 has length at least n, and the spanning tree on P_2 (and also P_1) has length at least $n-1$. Thus, the total length of $\mathcal{A}(I)$ or $\mathcal{A}(I')$ is at least $2n-1+c\cdot n\sqrt{n}$, for some $c > 0$, and an EMST has length at most $2n-2+n = 3n-2$, resulting in an approximation factor of

$$\frac{2n-1+c\cdot n\sqrt{n}}{3n-2} \geq \frac{2n-1+c\cdot n\sqrt{n}}{3n} = \frac{2}{3} - \frac{1}{3n} + \frac{c\cdot\sqrt{n}}{3} \geq x\sqrt{n} \quad \text{for } x = \frac{c}{3} \;\&\; n \geq \frac{1}{2}. \blacktriangleleft$$

As Lemma 2.3 applies to any pair out of I_a, I_b, I_c and I_d, we get the following corollary.

▶ **Corollary 2.4.** *If instances $I, I' \in \{I_a, I_b, I_c, I_d\}$, with $I \neq I'$, are mapped by $\mathcal{A}$ such that $\mathcal{A}(I) = \mathcal{A}(I')$, then $\mathcal{A}(I)$ or $\mathcal{A}(I')$ is an $\Omega(\sqrt{n})$-approximation of the EMST.*

Finally, we consider the whole boundary δD and prove that either $\mathcal{A}$ violates injectivity for parts of δD, or $\mathcal{A}$ maps δD around a hole in $\mathcal{S}$ (a cycle in the flip graph), in which case we find a contradiction with $\mathcal{A}$ being a continuous function. For the latter case of this proof, we consider the *fundamental group* of $\mathcal{S}$. The fundamental group of a graph is a basic concept in computational topology; essential concepts about this group are found in Section 4.3 of [3].

Proof sketch for Theorem 1.1. Consider the boundary δD of our input instances D. By Corollary 2.4, $\mathcal{A}$ maps the instances $I_a, I_b, I_c, I_d \in \delta D$ to different solutions in $\mathcal{S}$ or the theorem holds. Hence, we assume that I_a, I_b, I_c, I_d map to different solutions in $\mathcal{S}$.

Remember, $\mathcal{T}_S$ is defined by a flip graph, and we consider $\mathcal{S}$ to be the corresponding simplicial 1-complex. We make a case distinction on whether certain paths of δD are mapped injectively to $\mathcal{S}$ or not; see Figure 4. In the first two cases, Lemma 2.3 directly applies.

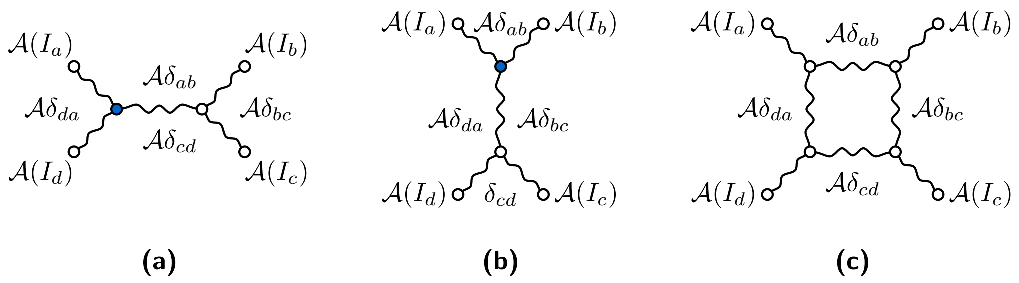


Figure 4 Cases for the mapping of δD by $\mathcal{A}$. Opposite paths share (blue) a vertex, **(a)** on $\mathcal{A}\delta_{ab}$ and $\mathcal{A}\delta_{cd}$, or **(b)** on $\mathcal{A}\delta_{bc}$ and $\mathcal{A}\delta_{da}$. Otherwise, **(c)** $\mathcal{A}\delta_{ab} \cap \mathcal{A}\delta_{cd} = \emptyset$ and $\mathcal{A}\delta_{bc} \cap \mathcal{A}\delta_{da} = \emptyset$.

Finally, we consider the case in which both $\mathcal{A}\delta_{ab} \cap \mathcal{A}\delta_{cd} = \emptyset$ and $\mathcal{A}\delta_{bc} \cap \mathcal{A}\delta_{da} = \emptyset$. We use a topological argument to derive a contradiction in this case: We consider two paths f and g starting at point $u \in \delta_{da}$ and ending at point $v \in \delta_{bc}$ and use the fact that f and g are homotopic; see Figure 5a. Since $\mathcal{A}$ is a continuous function, $\mathcal{A}f$ and $\mathcal{A}g$ are also homotopic.

We then consider the fundamental group of $\mathcal{S}$ to describe loops ℓ_f and ℓ_g that represent $\mathcal{A}f$ and $\mathcal{A}g$, respectively. We define a deformation retract of both $\mathcal{A}f$ and $\mathcal{A}g$ that removes any "unnecessary" spikes: essentially all parts of $\mathcal{A}f$ and $\mathcal{A}g$ that double back on themselves are removed; see Figure 5b. We show that this removal corresponds to simplifying the algebraic descriptors of ℓ_f and ℓ_g by removing *adjacent inverses*.

We want to repeatedly remove spikes from $\mathcal{A}f$ and $\mathcal{A}g$ until we find the *base paths* $F, G\colon [0,1] \to \mathcal{S}$, respectively, that connect $\mathcal{A}(u)$ to $\mathcal{A}(v)$ and cannot be shortened anymore using this operation. Equivalently, we can repeatedly check for adjacent inverses in the descriptors of ℓ_f and ℓ_g, that correspond to $\mathcal{A}f$ and $\mathcal{A}g$, until we find descriptors ℓ_F and ℓ_G, respectively, which correspond to loops that contain the base path (F and G, respectively) from $\mathcal{A}(u)$ to $\mathcal{A}(v)$. Note that $\mathcal{A}(u)$ may be the end of a spike that could be retracted. To prevent this, we apply a technical trick, to ensure that F and G must keep $\mathcal{A}(u)$ and $\mathcal{A}(v)$ as their endpoints. Even with this trick, our removal of adjacent inverses corresponds to a deformation retract, which is a homotopy. As homotopies are equivalence relations, $\mathcal{A}f$, $\mathcal{A}g$, F, and G are homotopic. Similarly, ℓ_F and ℓ_G are homotopic to ℓ_f and ℓ_g, respectively.

We are now ready to derive a contradiction by making a case distinction on whether the descriptors of ℓ_F and ℓ_G are exactly the same or not. When ℓ_F and ℓ_G have different descriptors, then by definition, F and G cannot be homotopic, leading to a contradiction.

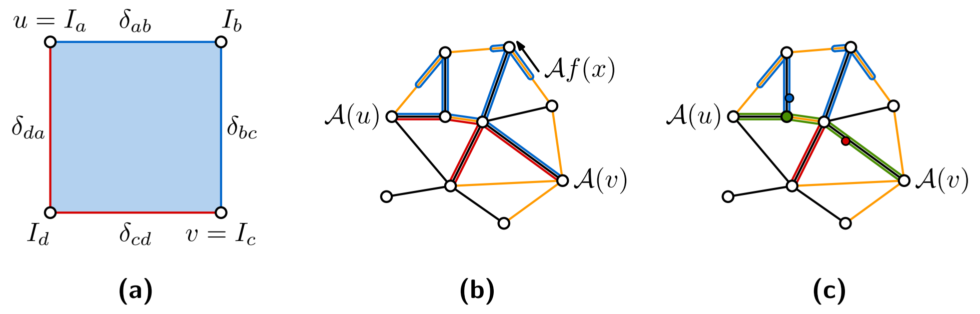


Figure 5 **(a)** Homotopic paths f (blue) and g (red). **(b)** The mappings $\mathcal{A}f$ (blue) and $\mathcal{A}g$ (red). The edges of the graph S are partitioned into spanning tree T (black) and generator edges (yellow). The point $(\mathcal{A}f)(x)$ can be retracted. **(c)** Base path $F = G$ is draw green. The blue vertex shows $\mathcal{A}(I_b)$; the red vertex shows $\mathcal{A}(I_d)$. Since $\mathcal{A}(I_b) \notin F$, the green vertex is closest to $\mathcal{A}(I_b)$ on $F = G$.

Thus, consider the case in which the descriptors of ℓ_F and ℓ_G are exactly the same, and hence $\mathcal{A}f$ and $\mathcal{A}g$ can both retract to the same base path $F = G$; see Figure 5c. In this case, (parts of) $\delta_{ab}, \delta_{bc}, \delta_{cd}$ and δ_{da} are mapped to the base path: when we traverse the base path from $\mathcal{A}(u)$ to $\mathcal{A}(v)$, we start by traversing $\mathcal{A}\delta_{ab}$ and $\mathcal{A}\delta_{da}$, and end by traversing $\mathcal{A}\delta_{bc}$ and $\mathcal{A}\delta_{cd}$. At some point during this traversal $\mathcal{A}\delta_{ab}$ swaps to $\mathcal{A}\delta_{bc}$, or $\mathcal{A}\delta_{da}$ swaps to $\mathcal{A}\delta_{cd}$, and at that point $\mathcal{A}\delta_{bc} \cap \mathcal{A}\delta_{da} \neq \emptyset$ or $\mathcal{A}\delta_{ab} \cap \mathcal{A}\delta_{cb} \neq \emptyset$, respectively, leading to a contradiction. ◀

3 Conclusion

We proved a lower bound on ρ_{TS} for stateless kinetic EMSTs using a combination of geometric and topological arguments. Interestingly, our proof relies entirely on the geometry of the considered input instances and on the fact that the topology of the solution space resembles a flip graph. However, note that the exact structure of the flip graph is irrelevant; it needs to be only connected. Our result hence extends to any connected flip graph for EMSTs.

References

1 Naoki Katoh, Takeshi Tokuyama, and Kazuo Iwano. On minimum and maximum spanning trees of linearly moving points. *Discrete & Computational Geometry*, 13(2):161–176, 1995.

2 Wouter Meulemans, Bettina Speckmann, Kevin Verbeek, and Jules Wulms. A framework for algorithm stability. In *Proc. 13th Latin American Symposium on Theoretical Informatics (LATIN)*, pages 805–819, 2018.

3 Wouter Meulemans, Bettina Speckmann, Kevin Verbeek, and Jules Wulms. A framework for algorithm stability. *CoRR*, abs/1704.08000, 2025.

4 Clyde Monma and Subhash Suri. Transitions in geometric minimum spanning trees. *Discrete & Computational Geometry*, 8(3):265–293, 1992.

5 Zahed Rahmati, Mohammad Ali Abam, Valerie King, Sue Whitesides, and Alireza Zarei. A simple, faster method for kinetic proximity problems. *Computational Geometry: Theory and Applications*, 48(4):342–359, 2015.

Disc Placement for Dynamic Bubble Charts

Annika Bonerath[1], William Evans[2], Jan-Henrik Haunert[1], David Kirkpatrick[2], and Stephen Kobourov[3]

1 University of Bonn, Germany
bonerath@igg.uni-bonn.de, haunert@igg.uni-bonn.de

2 University of British Columbia, Vancouver, Canada
will@cs.ubc.ca, kirk@cs.ubc.ca

3 Technical University Munich, Germany
stephen.kobourov@tum.de

Abstract

Static and dynamic bubble charts are used to visualize numerical data associated with geospatial locations. As overlapping discs affect readability and interpretability, we consider the problem of carefully reducing overlaps by displacement. We show that the underlying problem is NP-hard and propose approaches based on convex and non-convex quadratic programming. We evaluate our approaches on real-world data and provide a functional prototype.

1 Introduction

In this paper, we consider time-series data that is associated with a fixed spatial location, e.g., passenger numbers at different airports over time, or the population of cities over time. Bubble charts are often used to visualize such data: the position of each disc corresponds to the data point location and the radius encodes the statistic of interest [6, 8, 15]. A well-known problem of bubble charts is that discs obscure each other [2, 7, 9]. Examples of strategies to tackle this problem are: to scale down the discs which might lead to non-perceivable discs, to use transparent discs which can lead to high visual clutter [13], or to optimize the stacking-order [2] where still big parts of the discs' interior can be obscured. Our strategy is disc displacements. Removing all overlaps by displacement can lead to unclear associations between discs and data points [1, 3] and large parts of the background getting occluded. We do not enforce an overlap-free displacement, but optimize it with multiple criteria for the static and dynamic case, i.e., the discs can change their position and radius over time.

The input is a time series of T epochs and n data points where every data point $1 \leq i \leq n$ has for every time epoch $1 \leq t \leq T$ a 2D location $p_{i,t}$, which we call its *pinning point*, and a radius $r_{i,t}$. We denote a disc of data point $1 \leq i \leq n$ and time epoch $1 \leq t \leq T$ by $D_{i,t}$ with

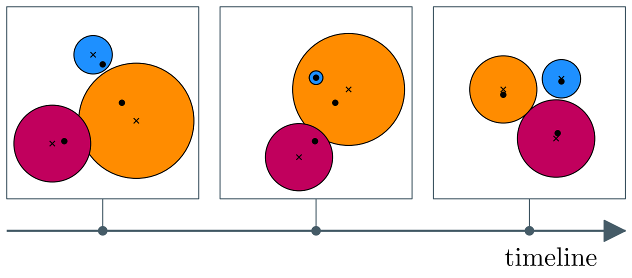


Figure 1 Dynamic problem setting: each box captures a moment in time, black dots correspond to the pinning points, crosses to disc centers.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.

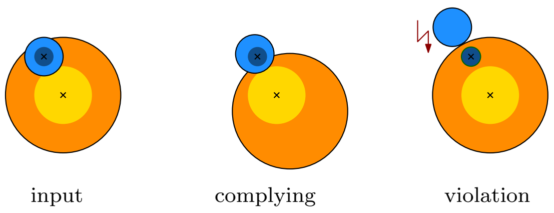


Figure 2 We set $\xi = 0.5$ and display the region with distance ξ times the disc radius in yellow for the orange disc and in dark blue for the blue disc.

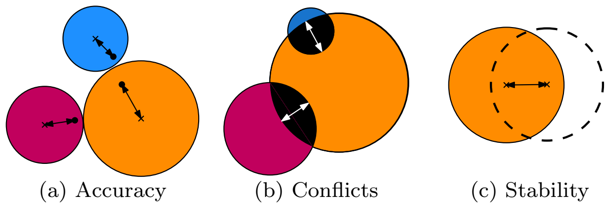


Figure 3 Our objectives. (b) Lenses (black) occur when discs are in conflict and the arrows depict the lenses' widths that are part of our objective. (c) We display the disc of time epoch t with solid lines and for time epoch $t+1$ with dashed lines.

disc center $c_{i,t}$ where the radius is $r_{i,t}$; see Figure 1. To allow a clear association, we want to find disc centers for every data point and time epoch where the distance between the center of each disc and its pinning point is at most ξ times its radius, where $0 < \xi \leq 1$; see Figure 2.

$$||\overrightarrow{c_{i,t}p_{i,t}}|| \leq \xi \cdot r_{i,t}. \tag{1}$$

We call this constraint *Pinning*. Amongst all disc placements that respect *Pinning*, we aim for three optimization dimensions, as illustrated in Figure 3.

- *Accuracy*: We want to minimize the sum of the squared Euclidean distances between the pinning point and disc center after displacement.

$$f_{Accuracy} = \sum_{i=1}^{n} \sum_{t=1}^{T} ||\overrightarrow{c_{i,t}p_{i,t}}||^2. \tag{2}$$

- *Conflict*: We want to minimize the lenses obtained by the intersection of two discs which we mathematically describe by the difference of the squared sum of the radii and the squared distance between the disc centers.

$$f_{Conflict} = \sum_{D_{i,t}, D_{k,t}} \max\left(0, (r_{i,t} + r_{k,t})^2 - ||\overrightarrow{c_{i,t}c_{k,t}}||^2\right) \tag{3}$$

- *Stability*: We want to minimize the sum of the squared Euclidean distances between the displacement of two consecutive time epochs.

$$f_{Stability} = \sum_{i=1}^{n} \sum_{t=1}^{T-1} ||\overrightarrow{d_{i,t+1}d_{i,t}}||^2. \tag{4}$$

Overall, we model our objective f as a weighted sum of the three criteria with balancing factors ρ_1, ρ_2, ρ_3 as follows:

$$f = \rho_1 \cdot f_{Accuracy} + \rho_2 \cdot f_{Conflict} + \rho_3 \cdot f_{Stability} \rightarrow \min. \tag{5}$$

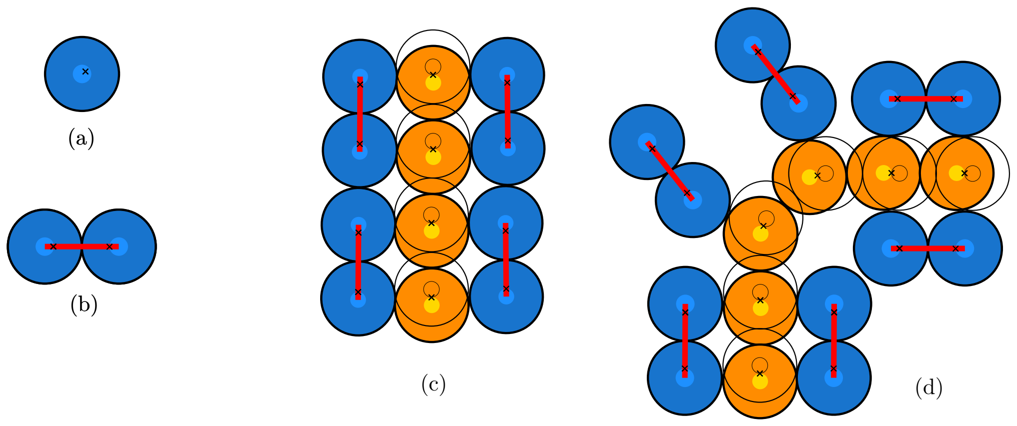


Figure 4 Basic components in the NP-hardness construction, with × marking pinning points. (a) Unit disc with blue "core" of radius ξ. (b) Forced pair: the only way to place the discs given the pinning points and core size is as illustrated. For (c)+(d) we illustrate two placements of non-intersecting discs within strategically placed forced pairs (blue): one version uses orange discs, the other discs with transparent filling. (c) Straight channel segment (blue) that forces the propagation of orange discs downwards (and transparent discs upwards). (d) Bent channel segment (blue) that forces the propagation of orange discs to the left and down (and transparent discs up and right).

2 Problem Complexity

In the following, we show that the decision problem "Does the given input have a solution without overlaps with bounded displacement" is NP-hard, even in the static case when all discs are the same (unit) size and the allowed displacement from the target location is the same for all discs. Note that this decision problem captures the extreme case of minimizing *Conflict* eq. (3). Its hardness implies that the optimization problem is also NP-hard.

▶ **Theorem 2.1.** *Deciding whether n unit discs can be placed without overlaps, while ensuring that each disc center is within distance at most ξ of its given target location is NP-hard, for any fixed ξ, where $0 < \xi < 1$.*

Proof. The proof relies on a polynomial-time reduction from a variant of the classic NP-hard problem 3-SAT. An instance of the *3-bounded Planar 3-SAT* problem [10] is a boolean formula ϕ in conjunctive normal form (CNF) such that: (i) each clause has at most 3 literals, (ii) each variable appears as a literal in exactly 3 clauses, and (iii) the associated graph $G(\phi)$ is planar. Given a formula ϕ with variables X and clauses C, the associated graph $G(\phi)$ has vertex set $X \cup C$ and edges $\{(x_i, c_j) \mid \text{ clause } c_j \text{ contains } x_i \text{ or } \overline{x_i}\}$.

Next, we sketch how to convert an instance ϕ of the *3-bounded Planar 3-SAT* problem into an instance $I(\phi)$ of unit discs, each with a specified target location, and a parameter ξ, such that ϕ is satisfiable if and only if $I(\phi)$ can be realized without disc overlaps where each disc center is at most distance ξ from its target location. In particular, using the basic components illustrated in fig. 4, we show how to model: (i) each variable with a fixed set of target disc locations (a *variable gadget*), (ii) each clause with a fixed set of target disc locations (a *clause gadget*), and (iii) directed "wires" of target disc locations connecting variable gadgets to clause gadgets.

Wires are built by concatenating straight and bent channel components. A wire in in a *charged* state if it forces the propagation of some disc color in its specified direction (and *uncharged* otherwise). The clause gadget (fig. 6) has one connection port for each of its three

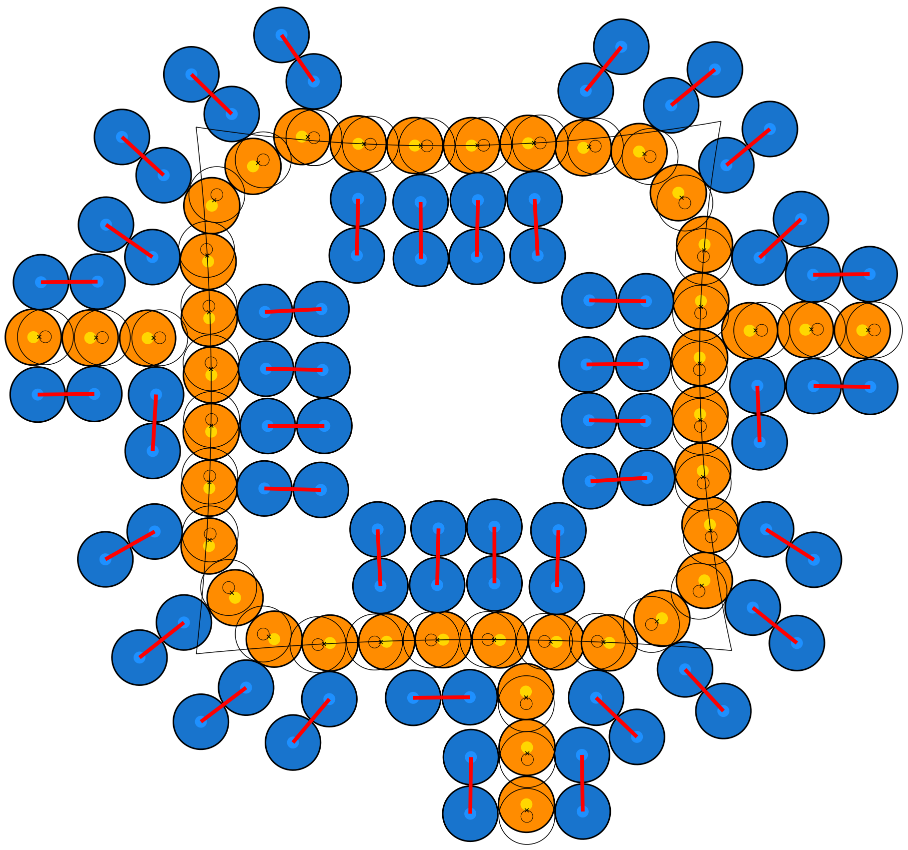


Figure 5 Variable gadget. The ring of discs can be in one of two placements (orange or transparent). Wires are shown connected to positive ports on the west and south sides and negative port on the east side.

literals, to which a wire originating at the variable gadget of the corresponding variable is attached. (If a clause has only two literals one of these ports is blocked as it would be with a charged wire attachment.) The gadget allows a realization without overlap if and only if at least one of the wires attached to its connection ports is uncharged.

The variable gadget (fig. 5) is a large square ring (with slightly concave sides) bounded by a collection of forced pairs. The discs between these forced pairs can be in one of two states, all orange (corresponding to `true`) or all transparent (corresponding to `false`). These inside discs are exposed on all four sides of the gadget, permitting the attachment of a wire on each side at either a positive or a negative *connection port*. If the variable gadget is in state `true`, then a wire attached to a negative connection port is unavoidably charged. Similarly, if the variable gadget is in state `false`, then a wire attached to a negative connection port is unavoidably charged. Otherwise, the attached wire could be uncharged.

Thus, if a wire is connected on one end to a positive (resp. negative) port on a variable gadget associated with variable x, and on the other end to one of the three ports of a clause gadget whose associated clause contains the literal x (resp. $\overline{x}$), then a positive (resp. negative) truth assignment to the variable gadget will permit a realization of the clause gadget that is free of overlap.

$\Rightarrow$) By construction, a satisfying assignment of the boolean formula ϕ allows for a realization of $I(\phi)$ with bounded displacement unit discs placed without overlaps, by (i)

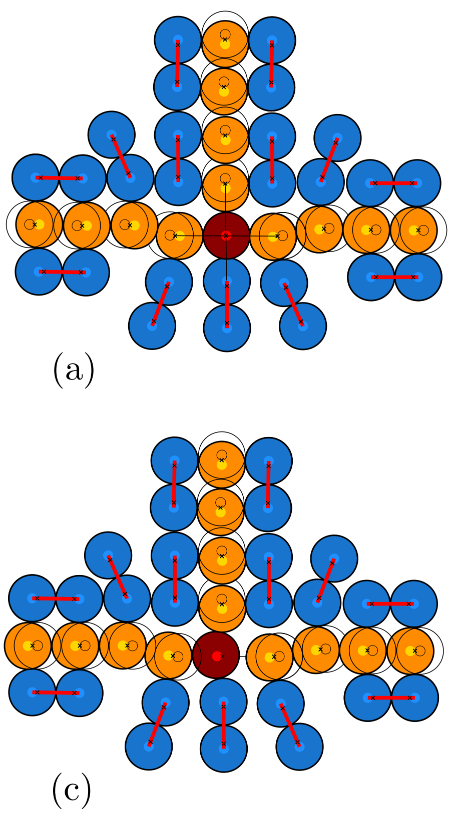


Figure 6 (a) clause gadget in state with three charged incoming wires. (b) ((c) and (d)) the same gadget with uncharged wires attached at the north (west and east) connection ports.

selecting the orange discs for each `true` variable gadget and the transparent discs for each `false` variable gadget, (ii) selecting an uncharged state for all wires connecting to positive ports on `true` variable gadgets, and all wires connecting to negative ports on `true` variable gadgets (all other wires are necessarily charged), and (iii) selecting a overlap-free placement for every clause gadget that takes advantage of the freedom associated with one of its uncharged attachments..

$\Leftarrow$) In the opposite direction, suppose $I(\phi)$ can be realized with bounded displacement unit discs placed without overlaps. Choose any such realization that maximizes the total number of wires in an uncharged state. Furthermore, for every variable gadget, either (i) all attached wires that are uncharged (resp. charged) attach at positive (resp. negative) ports, in which case the variable gadget must be in a `true` realization, or (ii) all attached wires that are uncharged (resp. charged) attach at negative (resp. positive) ports, in which case the variable gadget must be in a `false` realization. In either case, we can extract the assignment of the variables of ϕ by examining each variable gadget.

Since each clause is realized without overlapping discs, each clause gadget has at least one port with an uncharged attached wire, and hence at least one literal that is satisfied by the associated truth assignment. So the entire CNF formula ϕ evaluates to `true`.

It remains to show that the conversion of ϕ into $I(\phi)$ can be accomplished in polynomial time. Recall that the associated graph $G(\phi)$ has vertex set $V = X \cup U$, so it is linear in the size of the input ϕ. Note that the number of edges is at most $1.5|V|$, as variable vertices have degree 3 and clause vertices degree at most 3. It is known that any max-degree-3 n-vertex planar graph has an orthogonal drawing on an $n/2 \times n/2$ grid with at most one bend per edge (every vertex is at integer coordinates and every edge is axis-aligned with at most one turn) [4]. Starting with such an embedding of $G(\phi)$ we can increase the size of the grid to allow the realization of $G(\phi)$ with unit discs, replacing vertices of $G(\phi)$ with vertex gadgets and clauses gadgets, and edges of $G(\phi)$ with wire gadgets. This requires only a constant refinement of the underlying grid to provide sufficient space to avoid overlap.

There are two other concerns that need to be addressed: (i) as things are illustrated, it is not clear that it is possible to restrict target locations for disc centers to points on some suitably refined grid, and (ii) even if this is the case, it is necessary to argue that wire lengths that accommodate the placement of many disjoint unit discs can be chosen in a way that is compatible with the layout of $G(\phi)$. Both of these can be addressed by observing that our constructions do not rely in an essential way on the abutment of discs that is apparent in our figures. In particular, despite the resulting lack of rigidity in our constructions, all of our arguments would continue to hold, using the same target points, if the disc radius is reduced to $1-\varepsilon$, and ξ is replaced by $\xi + \varepsilon/2$ where ε is some sufficiently small positive constant. Thus, there is a reduction from *3-bounded Planar 3-SAT* to a stronger disc placement problem in which discs are required to be separated by at least ε and each disc is constrained to be placed within distance $\xi + \varepsilon/2$ of its target position. This latter problem is equivalent to a ε-separated disc placement problem in which each disc is constrained to be placed within distance ξ of its ε-perturbed target position. Of course, ε-perturbation of targets is sufficient to move them to grid positions on some suitably refined grid. ◀

In the theorem above we have assumed that the target locations are properly inside their corresponding discs. If this is not the case, Strijk et al. showed that given a set of points in the plane, placing a unit disc for each point such that the disc boundary contains the point and no two discs overlap is NP-hard [14].

3 Non-Convex Mathematical Programming Formulation

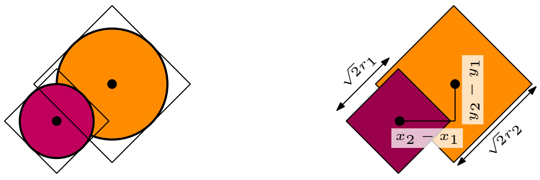
As shown in Section 2, the underlying problem is NP-hard. We present a mathematical programming formulation. While the properties *Accuracy*, *Stability*, and *Pinning* can be formulated as an efficiently-solvable convex formulation [5], the property *Conflict* leads to a non-convex formulation. In Section 4, we will present an efficient heuristic where we replace the non-convex formulation of *Conflict* with a convex formulation.

Let $p_{i,t} = (X_{i,t}, Y_{i,t})$ and $c_{i,t} = (x_{i,t}, y_{i,t})$. To model the displacements, we introduce variables for the disc centers $\mathtt{x}_{i,t}, \mathtt{y}_{i,t} \in [-\infty, \infty]$ and variables for the distance of disc centers and pinning points $\mathtt{dx}_{i,t}, \mathtt{dy}_{i,t} \in [-\infty, \infty]$ for $1 \le i \le n$ and $1 \le t \le T$. We introduce the following constraints, to assign variables correctly and while respecting *Pinning*.

$$\mathtt{x}_{i,t} = X_{i,t} + \mathtt{dx}_{i,t} \quad \text{and} \quad \mathtt{y}_{i,t} = Y_{i,t} + \mathtt{dy}_{i,t} \quad \text{and} \quad \mathtt{dx}_{i,t}^2 + \mathtt{dy}_{i,t}^2 \le \xi^2 r_{i,t}^2 \tag{6}$$

For *Conflict*, we introduce a cost variable $\mathtt{c}_{i,k,t}$ for every pair of discs $(D_{i,t}, D_{k,t})$.

$$\mathtt{c}_{i,k,t} \ge (r_{k,t} + r_{i,t})^2 - (\mathtt{x}_{i,t} - \mathtt{x}_{k,t})^2 - (\mathtt{y}_{i,t} - \mathtt{y}_{k,t})^2 \qquad \text{and} \qquad \mathtt{c}_{i,k,t} \ge 0 \tag{7}$$



(a) circumscribed diamonds (b) costs for overlap (push apart)

Figure 7 Convex formulation of conflicts with circumscribed diamonds.

Then, the formulation of *Conflict*, *Accuracy*, and *Stability* is straightforward.

$$f_{\mathit{Conflict}} = \sum_{i=1}^{n} \sum_{k=i+1}^{n} \sum_{t=1}^{T} \mathtt{c}_{i,k,t} \tag{8}$$

$$f_{\mathit{Accuracy}} = \sum_{i=1}^{n} \sum_{t=1}^{T} \mathtt{dx}_{i,t}^2 + \mathtt{dy}_{i,t}^2 \tag{9}$$

$$f_{\mathit{Stability}} = \sum_{i=1}^{n} \sum_{t=1}^{T-1} (\mathtt{dx}_{i,t} - \mathtt{dx}_{i,t+1})^2 + (\mathtt{dy}_{i,t} - \mathtt{dy}_{i,t+1})^2 \tag{10}$$

Overall, given ξ, and ρ_1, ρ_2, and ρ_3, we obtain the objective by Equation (5).

4 Heuristic: Convex Mathematical Programming Formulation

As expected, preliminary experiments with the non-convex program showed that this approach is not suitable for real-world data sets. We suggest to solve our problem heuristically by replacing each disc with its *circumscribed diamond* that we define as the bounding square of a disc with side-length equal to the diameter of the disc, and which is rotated by 45°; Figure 7. This approach is adapted from Niedermann and Haunert [12] and Meulemans [11].

To allow a convex formulation, we enforce that the initial east-west and north-south order of the pinning points is preserved after the displacement. More formally, for every pair of discs $D_{i,t}$ and $D_{k,t}$ that could be in conflict after the displacement, we introduce the following two linear constraints:

$$\text{if } X_{i,t} \le X_{k,t}\text{:} \quad \mathtt{x}_{i,t} \le \mathtt{x}_{k,t} \quad \text{and otherwise:} \quad \mathtt{x}_{k,t} \le \mathtt{x}_{i,t} \tag{11}$$

$$\text{if } Y_{i,t} \le Y_{k,t}\text{:} \quad \mathtt{y}_{i,t} \le \mathtt{y}_{k,t} \quad \text{and otherwise:} \quad \mathtt{y}_{k,t} \le \mathtt{y}_{i,t}. \tag{12}$$

With the enforced order, we can formulate *Conflict* as a linear constraint. Let $D_{i,t}$ and $D_{k,t}$ be two discs that could possibly be in conflict after displacement with $r_{i,t} < r_{k,t}$. Let $\mathtt{c}' \in \mathbb{R}$ be a variable that represents the costs for *Conflict*, i.e., the width of the smaller side of the rectangle that is derived by the intersection of the boundaries of the circumscribed diamonds. We introduce the constraints depending on the east-west and north-south order. In the following, we give the constraints for $X_{i,t} \le X_{k,t}$ and $Y_{i,t} \le Y_{k,t}$; see Figure 7.

$$\mathtt{c}'_{i,k,t} \ge \sqrt{2}(r_{k,t} + r_{i,t}) - (\mathtt{x}_{k,t} - \mathtt{x}_{i,t}) - (\mathtt{y}_{k,t} - \mathtt{y}_{i,t}) \quad \text{and} \quad \mathtt{c}'_{i,k,t} \ge 0 \tag{13}$$

The other three cases can be derived analogously. Then, we can replace f_{Conflict} by:

$$f'_{\mathit{Conflict}} = \sum_{i=1}^{n} \sum_{k=i+1}^{n} \sum_{t=1}^{T} \mathtt{c}_{i,k,t}. \tag{14}$$

5 Experiments

We performed experiments on different data sets and compared the two presented approaches. Results can be found under `http://www2.geoinfo.uni-bonn.de/html/visualization/bubblecharts/`.

6 Outlook

It would be interesting whether one can show an approximation factor for the diamond approximation. Further, we plan to extend this work by introducing other algorithmic approaches and analyze this project from a perceptual point of view.

References

1 Ken Been, Martin Nöllenburg, Sheung-Hung Poon, and Alexander Wolff. Optimizing active ranges for consistent dynamic map labeling. In *Proceedings of Symposium on Computational geometry*, pages 10–19, 2008. `doi:https://doi.org/10.1145/1377676.137768`.

2 Sergio Cabello, Herman Haverkort, Marc Van Kreveld, and Bettina Speckmann. Algorithmic aspects of proportional symbol maps. *Algorithmica*, 58:543–565, 2010. `doi:https://doi.org/10.1007/11841036_64`.

3 Anne P Hillstrom, Hannah Wakefield, and Helen Scholey. The effect of transparency on recognition of overlapping objects. *Journal of Experimental Psychology: Applied*, 19(2):158, 2013. `doi:10.1037/a0033367`.

4 Goos Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16:4–32, 1996. `doi:https://doi.org/10.1007/BF02086606`.

5 M.K. Kozlov, S.P. Tarasov, and L.G. Khachiyan. The polynomial solvability of convex quadratic programming. *USSR Computational Mathematics and Mathematical Physics*, 20(5):223–228, 1980. URL: `https://www.sciencedirect.com/science/article/pii/0041555380900981`, `doi:https://doi.org/10.1016/0041-5553(80)90098-1`.

6 Guilherme Kunigami, Pedro J de Rezende, Cid C de Souza, and Tallys Yunes. Optimizing the layout of proportional symbol maps: Polyhedra and computation. *INFORMS journal on computing*, 26(2):199–207, 2014.

7 David Lamb. An automated displaced proportional circle map using delaunay triangulation and an algorithm for node overlap removal. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 52(4):364–370, 2017. `doi:10.3138/cart.52.4.2016-0007`.

8 Zeyu Li, Ruizhi Shi, Yan Liu, Shizhuo Long, Ziheng Guo, Shichao Jia, and Jiawan Zhang. Dual space coupling model guided overlap-free scatterplot. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):657–667, 2023. `doi:10.1109/TVCG.2022.3209459`.

9 Haipeng Liu, Ling Zhang, Yi Long, and Yi Zheng. Real-time displacement of point symbols based on spatial distribution characteristics. *ISPRS International Journal of Geo-Information*, 8(10):426, 2019. `doi:https://doi.org/10.3390/ijgi8100426`.

10 Ján Maňuch and Daya Ram Gaur. Fitting protein chains to cubic lattice is NP-complete. *Journal of bioinformatics and computational biology*, 6(01):93–106, 2008. `doi:10.1142/s0219720008003308`.

11 Wouter Meulemans. Efficient optimal overlap removal: Algorithms and experiments. *Computer Graphics Forum*, 38:713–723, 2019. `doi:https://doi.org/10.1111/cgf.13722`.

12 Benjamin Niedermann and Jan-Henrik Haunert. Focus+context map labeling with optimized clutter reduction. *International Journal of Cartography*, 5(2-3):158–177, 2019. `arXiv:https://doi.org/10.1080/23729333.2019.1613072`, `doi:10.1080/23729333.2019.1613072`.

13 Tomasz Opach, J Korycka-Skorupa, I Karsznia, T Nowacki, I Golebiowska, and JK Rod. Visual clutter reduction in zoomable proportional point symbol maps. *Cartography and Geographic Information Science*, 46(4):347–367, 2019. `doi:https://doi.org/10.1080/15230406.2018.1490202`.

14 Tycho Strijk and Alexander Wolff. Labeling points with circles. *International Journal of Computational Geometry & Applications*, 11(02):181–195, 2001. `doi:https://doi.org/10.1142/S0218195901000444`.

15 Mereke van Garderen, Barbara Pampel, Arlind Nocaj, and Ulrik Brandes. Minimum-displacement overlap removal for geo-referenced data visualization. *Computer Graphics Forum*, 36:423–433, 2017. `doi:https://doi.org/10.1111/cgf.13199`.

Parameterized Algorithms for Crossing Number with Forbidden Topological Crossing Patterns[*]

Miriam Münch[1] **and Ignaz Rutter**[1]

1 **Faculty of Computer Science and Mathematics, University of Passau, Germany**
`{muenchm,rutter}@fim.uni-passau.de`

Abstract

Beyond-planar graph classes are defined via forbidden crossing patterns in a drawing. In this work, we study *topological crossing patterns* that are specified by topological graphs that come with a fixed embedding. We give a parameterized algorithm that decides for any fixed family $\mathcal{F}$ of topological crossing patterns, whether a given graph G admits a drawing that avoids all patterns in $\mathcal{F}$ and that has at most c crossings in FPT time w.r.t. c. In particular, this shows that the weak and strong fan-planar crossing number as well as the 1-planar straight-line crossing number are FPT.

1 Introduction

Beyond-Planarity [10, 5] is a concept aimed at capturing classes of graphs that admit drawings where crossings are in some sense well-distributed. An example are *k-planar* graphs which admit a drawing where every edge has at most k crossings. We study the problem of testing whether a graph G admits a drawing with at most c crossings that satisfies the restriction $\mathcal{R}$. An interesting question is for which beyond-planar graph classes this problem is FPT with respect to c. Hamm and Hliněný [8] gave a positive answer for 1-planar graphs. A recent result of Hliněný and Colin de Verdière [4] gave FPT algorithms for various crossing number related problems such as the gap-planar crossing number.

Often the restriction $\mathcal{R}$ is expressed in terms of forbidden configurations of crossings that a drawing of G must avoid. Recently Münch and Rutter [11] proposed a formalization of such configurations as *combinatorial crossing patterns* and showed that, for any fixed family $\mathcal{F}$ of combinatorial crossing patterns, there is an algorithm that decides whether G admits a drawing with at most c crossings that avoids all patterns in $\mathcal{F}$ in FPT time with respect to c. A combinatorial crossing pattern is present if and only if it is contained as some kind of subgraph in the planarization of the drawing. While this suffices to capture a rich set of beyond-planar graph classes, there are beyond-planar graphs where this is too restricted. Two cases are weak/strong fan-planar graphs [1] and straight-line 1-planar graphs [12]. Both can also be described in terms of forbidden crossing configurations but the presence of a forbidden configuration in a drawing additionally requires that the embedding the drawing induces on the subgraph is the same as in the forbidden pattern. We propose a notion of topological crossing patterns and show the following meta theorem, which in particular implies that the beyond-planar crossing number is FPT for straight-line 1-planar graphs as well as for weak and strong fan-planar graphs.

▶ **Theorem 1.** *For any fixed family $\mathcal{F}$ of topological crossing patterns, the problem of testing whether a given graph G admits a drawing with at most c crossings that avoids all patterns in $\mathcal{F}$ is FPT with respect to c.*

[*] Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 541433306.

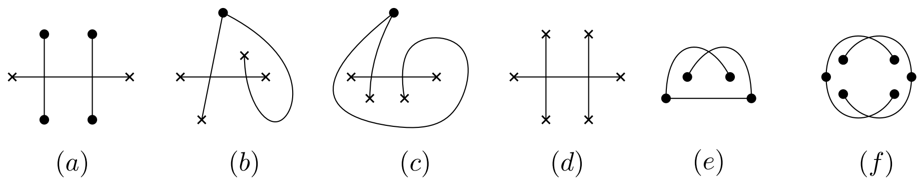


Figure 1 Examples of topological crossing patterns. Real vertices are depicted as disks, subdivision vertices as crosses.

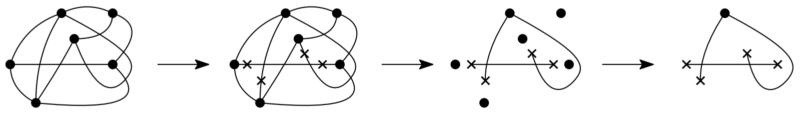


Figure 2 A topological graph G (left) that contains the topological pattern P shown in Figure 1d.

Our algorithm has two phases. First, we bound the treewidth of the input graph based on Grohe's FPT-algorithm for crossing number [7, 11]. Second, we show that the set of graphs that admit a planarization with at most c crossings that has an embedding that avoids all patterns in $\mathcal{F}$ is defineable in monadic second-order logic. The result then follows from Courcelle's Theorem [3]. We remark that Hamm, Klute and Parada [9] simultaneously and independently obtained an analogous result based on similar core ideas.

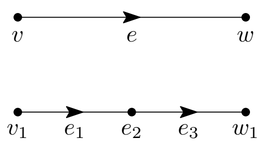
2 Topological Crossing Patterns

A *topological crossing pattern* is a topological graph $P = (V_P, E_P)$ with $V_P = R \cup S$ with a fixed outer face such that (i) each *subdivision vertex* in S has degree 1, (ii) each vertex is incident to a crossed edge, (iii) the set of *real vertices* R does not contain a cut-vertex v such that $P - v$ contains a connected component without crossing vertices and (iv) R does not contain a split-pair u, v such that $P - \{u, v\}$ contains a connected component without crossing vertices. Conditions (ii) – (iv) are necessary technical conditions but they are also natural assumptions as our crossing patterns shall express conditions on crossing configurations in drawings rather than structural conditions on the input graph.

Two topological crossing patterns are *isomorphic* if there is an isomorphism between them that maps real vertices to real vertices, subdivision vertices to subdivision vertices, and preserves the embedding including the crossings and the outer face. We say that a topological graph D *contains* a topological crossing pattern P if and only if a topological crossing pattern isomorphic to P can be obtained from D by subdividing edges with subdivison vertices as well as deleting edges and isolated vertices; see Figure 2 for an illustration. It can be shown that a drawing of a graph is (i) weakly fan-planar if and only if it avoids the patterns in Figure 1$(a), (b)$, (ii) strongly fan-planar if and only if it avoids the patterns in Figure 1$(a), (b), (c)$ [1] and (iii) straight-line 1-planar if and only if it avoids the patterns in Figure 1$(d), (e), (f)$ [12].

3 Relational Structures and Monadic Second-Order Logic

Let $\mathcal{R}$ be a finite set of relation symbols and let each $R \in \mathcal{R}$ be associated with an arity $\rho(R) \in \mathbb{N}$. An $\mathcal{R}$*-structure* is a tuple $S = \langle D_S, (R_S)_{R \in \mathcal{R}} \rangle$ where the *domain* D_S is a finite set and for each $R \in \mathcal{R}$, R_S is a $\rho(R)$-ary relation. An *ordered structure* $(X, \leq)$ is a structure X together with a linear ordering of its domain. Monadic second-order logic



$in_{1,1}(v,e) := in(v,e) \wedge e \notin F$ $out_{1,1}(v,e) := out(v,e)$

$in_{1,2}(v,e) := e \in F \wedge v = e$ $out_{2,3}(v,e) := e \in F \wedge v = e$

$in_{1,3}(v,e) := in(v,e) \wedge e \in F$

Figure 3 Illustration of a transduction that subdivides edges.

(MS for short) is the fragment of second-order logic that allows quantification over elements and over unary relations (i.e., subsets of the domain) but no quantification of arity greater than 1. For an $\mathcal{R}$-structure X and an MS-formula φ, we write $X \vDash \varphi$ if X is a model of φ. A subset L of the set of $\mathcal{R}$-structures is *MS-definable* if $L = \{X \mid X \vDash \varphi\}$ of a monadic second-order formula φ. An MS-formula φ is *order-invariant* if for any structure X and linear orderings $\leq, \leq'$ we have $(X, \leq) \vDash \varphi$ if and only if $(X, \leq') \vDash \varphi$.

A *graph* G is a relational structure $\langle V \cup E, in, out \rangle$ where $in, out \subseteq V \times E$ are the outgoing and the incoming incidence relation of G, respectively. A *planarization* P is a relational structure $\langle G, cont \rangle$ that consists of a graph G and a relation $cont \subseteq V \times E \times E$, where additionally some vertices may be labeled as *crossing vertices*. Such crossing vertices need to have in-degree 2 and out-degree 2 and *cont* describes for each crossing vertex v which pairs of in- and outgoing edges stem from the same original edge of the original graph. A *planar map* is a relational structure $\langle P, succ \rangle$, where P is a planarization and $succ \subseteq V \times E \times E$ encodes a planar rotation system of the graph P as a counterclockwise successor relation. A *plane map* $\langle M, v_o, e_o \rangle$ consists of a planar map together with two constants, a vertex v_o and an edge e_o incident to v_o, which together encode the outer face as the face that lies right of the edge e_o when traversing it from v_o.

An *MS-transduction* transforms a structure S with domain D_S into a structure T with domain $D_T \subseteq D_S \times \{0, \ldots, n\}$ for a fixed $n \in \mathbb{N}$. Transductions may have parameters that specify subsets of the domain. We now give an example of a transduction with parameter F that associates with a graph G the graph G' that we obtain from G by subdividing every edge in F; see Figure 3 for an illustration. For the domain of G' we use $(V \cup E) \times \{1\} \cup F \times \{2\} \cup F \times \{3\}$; i.e, the transduction is 3-copying. For vertices and edges not in F we use only the first copy. For each edge e in F we use the first and the third copy to represent the two edges into which e is subdivided, and we use the second copy of e as the subdivision vertex. The non-trivial relations corresponding to G' are given in Figure 3; all remaining ones are false.

The composition of two MS-transductions is again an MS-transduction. For every MS-transduction and every MS-formula φ, there exists a *backwards translation* of φ which is an MS-formula ψ with the property that if a structure T is defined from a structure S by the transduction, then $S \vDash \psi$ if and only if $T \vDash \varphi$ [3, Thm. 1.40]. For example, if φ defines a graph property, the transduction above implies that there exists a formula $\psi(F)$ with free variable F such that $G \vDash \psi(F)$ if and only if $G' \vDash \varphi$, where G' is obtained from G by subdividing the edges in F. In particular, the formula $\exists F \psi(F)$ expresses that a graph can be modified to satisfy the property φ by subdividing some of its edges once; i.e., these graphs are MS-definable.

For a plane map M and a fixed set of edges E' there is an MS-definable transduction that associates with M the planar map where every edge in E' is subdivided. Similarly there are MS-definable transductions that associate with M the planar map where a fixed set of edges and a fixed set of vertices is removed. Further, for any fixed plane map F the set of plane maps that are isomorphic to F is MS-definable. Together this shows the following.

▶ **Theorem 2.** *Let $\mathcal{F}$ be a family of topological crossing patterns. The set of plane maps that contains no pattern from $\mathcal{F}$ is MS-definable.*

4 Constructing a Plane Map from a Planarization

Let P be a planarization with at most c crossings. Our goal is to define a transduction Φ whose number of parameters is bounded in c and that associates with P an embedding $\mathcal{E}$ (encoded as a plane map). Due to space constraints, we only sketch the biconnected case.

A split component with respect to a vertex pair $\{u, v\}$ is *irrelevant* if it contains no crossing vertex. An edge is called *irrelevant* if it is contained in some irrelevant split component. The remaining edges are *relevant*. By property (*iv*), a topological crossing pattern that occurs in an embedding of P cannot contain an irrelevant edge. Thus we may arbitrarily re-embed the irrelevant split components. We call two planar embeddings $\mathcal{E}, \mathcal{E}'$ of P *equivalent* if for each vertex v of G, we have that the circular order of their incident relevant edges are the same. For an embedding $\mathcal{E}$, a vertex v and an edge e incident to v, we write $(\mathcal{E}, v, e)$ to denote the plane embedding of P whose outer face is the face that lies right of e seen from v.

▶ **Lemma 3.** *Let P be a planarization, let $\mathcal{E}, \mathcal{E}'$ be two equivalent planar embeddings of P and let (v, e) be a pair of a vertex v and a relevant edge e incident to v. Let f, f' be the faces that lie to the right of e when traversing it starting at v in $\mathcal{E}$ and $\mathcal{E}'$, respectively. Then, for any topological crossing pattern F, we have that F is contained in $(\mathcal{E}, f)$ if and only if it is contained in $(\mathcal{E}', f')$.*

We now express for each planar embedding $\mathcal{E}$ of P an equivalent embedding using sets with MS-definable properties whose number is bounded in c. Thus, by existentially quantifying these sets, for each planar embedding of P, we can obtain an equivalent one. Like Courcelle, whose work [2] is the basis of our construction, we need an auxiliary order $\leq$, which we only use for decisions that affect irrelevant edges. As a first step, we number the crossings in P by specifying c pairwise disjoint sets $C_1, \dots, C_c$, each of which contains at most one crossing vertex, such that $C_i = \emptyset$ implies $C_{i+1} = \emptyset$. We refer to the unique element in C_i as c_i (if it exists). We associate

- with each crossing vertex c_i, four sets of edges $E_i^1, \dots, E_i^4$ called the *rotation set* of c_i,
- with each pair (c_i, c_j) with $i < j$ of crossing vertices, a set $W_{i,j}$ of *path edges* that constitute a simple path $w_{i,j}$ from c_i to c_j, called the *witness path* of c_i and c_j, and a set $M_{i,j}$ of *R-marker edges*,
- with each partition $\mathcal{P}$ of a subset of the set C of crossing vertices, a *merge scheme*, which consists of sets $X_{\mathcal{P}}^1, \dots, X_{\mathcal{P}}^c$ of crossing vertices.

We call such an association an *embedding scheme* of $(P, \leq)$. We let Σ denote the set of all embedding schemes of $(P, \leq)$ and we let Ω denote all the planar embeddings of P.

▶ **Theorem 4.** *There is a relation $\Phi \subseteq \Sigma \times \Omega$ that (i) associates with each embedding scheme S of an ordered planarization $(P, \leq)$ at most one planar embedding in Ω and (ii) for each planar embedding $\mathcal{E} \in \Omega$ there exists an embedding $\mathcal{E}' \in \Omega$ that is equivalent to $\mathcal{E}$ and such that there exists an embedding scheme $S \in \Sigma$ with $(S, \mathcal{E}') \in \Phi$.*

Proof. Let $\mathcal{E}$ be a fixed embedding. We construct an embedding scheme that encodes an embedding equivalent to $\mathcal{E}$. Recall that in the biconnected case the embedding is determined by flipping R-nodes and ordering P-nodes.

First, let μ be a node of the SPQR-tree of P. If $\mathrm{sk}(\mu)$ contains a crossing vertex, then let c_i be the crossing vertex with the smallest index that is contained in $\mathrm{sk}(\mu)$. The rotation

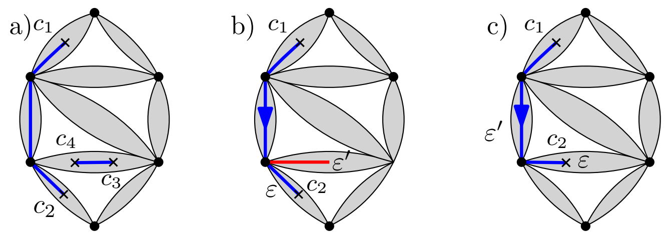


Figure 4 R-node: (a) separates (c_1, c_2) but not (c_3, c_4), (b,c) choice of marker edge (red).

of c_i uniquely determines the embedding sk(μ). We encode this rotation as four pairwise disjoint sets $E_1^i, \dots, E_4^i$, each containing exactly one edge incident to c_i according to the counterclockwise order around c_i. Observe that while c_i may decide the embeddings of more than one such block, they all associate the exact same sets with c_i.

If sk(μ) does not contain a crossing vertex, we distinguish cases based on the type of μ. For S-nodes there is no embedding decision. For a P-node consider the virtual edges that do not correspond to irrelevant split components. If there are at most two such virtual edges, then the embedding of the P-node is inconsequential for the rotation system of the relevant edges and we can simply order all virtual edges according to $\leq$. If there are at least three such components, the distribution of the crossing vertices to the virtual edges of sk(μ) defines a partition $\mathcal{P}_\mu$ of the crossing vertices. We encode the order of the virtual edges into a sequence of sets $X^1_{\mathcal{P}_\mu}, \dots, X^c_{\mathcal{P}_\mu}$, each of which is either empty or contains a distinct set from $\mathcal{P}_\mu$. The order of these corresponds to the left to right order in $\mathcal{E}$, starting with the parent edge if it contains a relevant edge. Moreover, we again require $X^i_{\mathcal{P}_\mu} = \emptyset \Rightarrow X^{i+1}_{\mathcal{P}_\mu} = \emptyset$. Observe that, from this information, we can fully recover the order of the relevant edges of the P-node and we can simply put the irrelevant virtual edges at the end, according to the order $\leq$. Moreover, observe that distinct nodes of the SPQR-tree yield distinct partitions. Therefore, the given information influences only the embedding choice of sk(μ).

We only sketch the R-node case, where the embedding is a binary decision. A pair of crossing vertices c_i, c_j is *separated by* μ if their witness path $w_{i,j}$ contains an edge e that belongs to a virtual edge of μ and that is incident to a vertex of sk(μ); see Fig. 4a. To encode the embedding, we take the last edge ε of sk(μ) that belongs to the witness path of the lexicographically smallest crossing pair that is separated by μ. We then take an edge from the edge ε' of $skel(\mu)$ that succeeds ε in counterclockwise order as marker edge (Figure 4b), unless ε' also belongs to the witness path (Figure 4c).

Clearly an embedding scheme constructed as described above encodes an embedding equivalent to ε, thus (ii) holds. Conversely for (i), it can be shown that there is an MS-formula that expresses that an embedding scheme has a form that is obtained from an embedding in the way described above. In the positive case, a unique embedding can be obtained from the embedding scheme as described above, where the order $\leq$ is used to order the irrelevant edges. ◀

An embedding scheme S is *valid* if $(S, \mathcal{E}) \in \Phi$ for some $\mathcal{E} \in \Omega$. In this case, we write $\Phi(S)$ for $\mathcal{E}$. All the conditions stated in the proof of Theorem 4 can be expressed in MS, thus by combining Theorem 4 with Courcelle's transduction [2] we obtain that Φ is indeed a transduction with parameter S. Let $\mathcal{F}$ be a fixed family of crossing patterns. By existentially quantifying the embedding scheme S and requiring that the transduced plane map avoids all crossing patterns in $\mathcal{F}$, which is MS-definable by Theorem 2, we obtain the following theorem. The order-invariance follows from Lemma 3 and the observation that a different order $\leq'$ yields an equivalent embedding.

▶ **Theorem 5.** *For any fixed family $\mathcal{F}$ of crossing patterns, the set of ordered planarizations $(P, \leq)$ with at most c crossings that admit an embedding that avoids all patterns in $\mathcal{F}$ is MS-definable by an order-invariant formula φ.*

Using transductions to add up to c crossings in a similar fashion to Grohe [7], the fact that order-independent model-checking is FPT with respect to treewidth [6] and a reduction that bounds the treewidth in terms of the crossing number [7, 11] yields Theorem 1.

References

1 Otfried Cheong, Henry Förster, Julia Katheder, Maximilian Pfister, and Lena Schlipf. Weakly and strongly fan-planar graphs. In *International Symposium on Graph Drawing and Network Visualization*, pages 53–68. Springer, 2023.

2 Bruno Courcelle. The monadic second-order logic of graphs XII: planar graphs and planar maps. *Theor. Comput. Sci.*, 237(1-2):1–32, 2000.

3 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012.

4 Éric Colin de Verdière and Petr Hlinený. FPT algorithms for crossing number problems: A unified approach. *CoRR*, abs/2410.00206, 2024.

5 Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. A survey on graph drawing beyond planarity. *ACM Computing Surveys*, 52(1):4:1–4:37, 2019.

6 Kord Eickmeyer, Jan van den Heuvel, Ken-Ichi Kawarabayashi, Stephan Kreutzer, Patrice Ossona De Mendez, Michał Pilipczuk, Daniel A. Quiroz, Roman Rabinovich, and Sebastian Siebertz. Model-checking on ordered structures. *ACM Transactions on Computational Logic*, 21(2):no.11, 1–28, 2020.

7 Martin Grohe. Computing crossing numbers in quadratic time. *Journal of Computer and System Sciences*, 68(2):285–302, 2004.

8 Thekla Hamm and Petr Hlinený. Parameterised partially-predrawn crossing number. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry (SoCG '22)*, volume 224 of *LIPIcs*, pages 46:1–46:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

9 Thekla Hamm, Fabian Klute, and Irene Parada. Computing crossing numbers with topological and geometric restrictions. *CoRR*, abs/2412.13092, 2024.

10 Seok-Hee Hong and Takeshi Tokuyama, editors. *Beyond Planar Graphs*. Springer Singapore, 2020.

11 Miriam Münch and Ignaz Rutter. Parameterized algorithms for beyond-planar crossing numbers. In *32nd International Symposium on Graph Drawing and Network Visualization, GD 2024*, volume 320 of *LIPIcs*, pages 25:1–25:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

12 Carsten Thomassen. Rectilinear drawings of graphs. *Journal of Graph Theory*, 12(3):335–341, 1988.

Computation of the canonical tree decomposition of a set of points in the plane into mutually avoiding parts

Emeric Gioan[1] and Yann Marin[2]

1 LIRMM, Université de Montpellier, CNRS, Montpellier, France
emeric.gioan@lirmm.fr

2 LIRMM, Université de Montpellier, CNRS, Montpellier, France
yann.marin@lirmm.fr

Abstract

This paper addresses the problem of efficiently decomposing a set of points in the plane into mutually avoiding parts, along with its abstraction in terms of chirotopes and oriented matroids. In [M. Bouvel, V. Féray, X. Goaoc and F. Koechlin. A canonical tree decomposition for chirotopes. Proceedings SoCG 2024] and [M. Bouvel, V. Féray, X. Goaoc and F. Koechlin. A canonical tree decomposition for order types, and some applications. arXiv:2403.10311, 2024], the authors introduced a specific tree decomposition of a rank-3 acyclic uniform oriented matroid (or, in particular, of a finite set of points in general position in the plane), by means of a geometric inductive construction. They raised the question of how to efficiently compute this decomposition tree. In this paper, we reformulate this decomposition in terms of the theory of tree representations of set families, especially of symmetric-crossing families. Using an appropriate data structure, we then answer the above question and give a method to calculate this tree in time $O(n^3)$ where n is the number of elements.

Related Version This paper is an extended abstract of the paper [8].

1 Introduction

This paper is based on notions addressed in the conference paper [2] and its full version [3].

Given a set of points P in the plane in general position (that is, no three of them are colinear), a bipartition $(A, \overline{A})$ of points of P is said to be **mutually avoiding** if no line spanned by two points of A separates two points of $\overline{A}$ and vice versa. In other words, no line spanned by two points of A intersects the convex hull of $\overline{A}$ and vice-versa. See Figure 1.

In [2, 3], they generalize the notion of mutually avoiding bipartitions to “rank-3 acyclic uniform oriented matroids” using the notion of “chirotopes” (more precisely, they use the notion of “order types” but we prefer to speak of oriented matroids). They introduce the notion of “(canonical) tree decomposition” of a set of points $P \subseteq \mathbb{R}^2$ in general position (labeled by a set E), or more generally of a rank-3 acyclic uniform oriented matroid M on a ground set E. The leaves of such a tree correspond to elements of E and every node is decorated with a smaller set of point, or oriented matroid, that is either convex or indecomposable (i.e., it has no mutually avoiding bipartition). From this tree, we can retrieve P, or M, and all its mutually avoiding bipartitions. (See Section 2 for details.)

▶ **Example 1.1.** Figure 2 depicts a configuration P of points in the plane. Figure 3 shows the tree decomposition of (the oriented matroid of) this set of points.

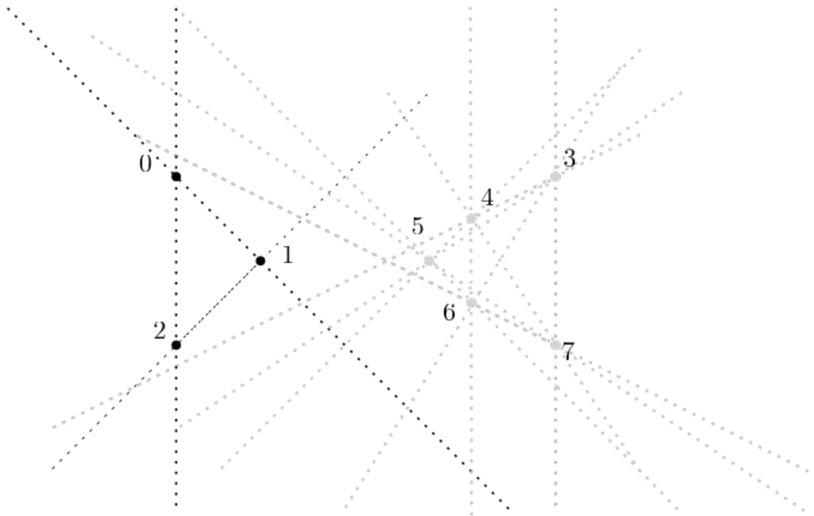


Figure 1 A set of eight points. No line spanned by two blacks points or two grays points separates two grays points or two black points, respectively. Such a bipartition is said to be mutually avoiding.

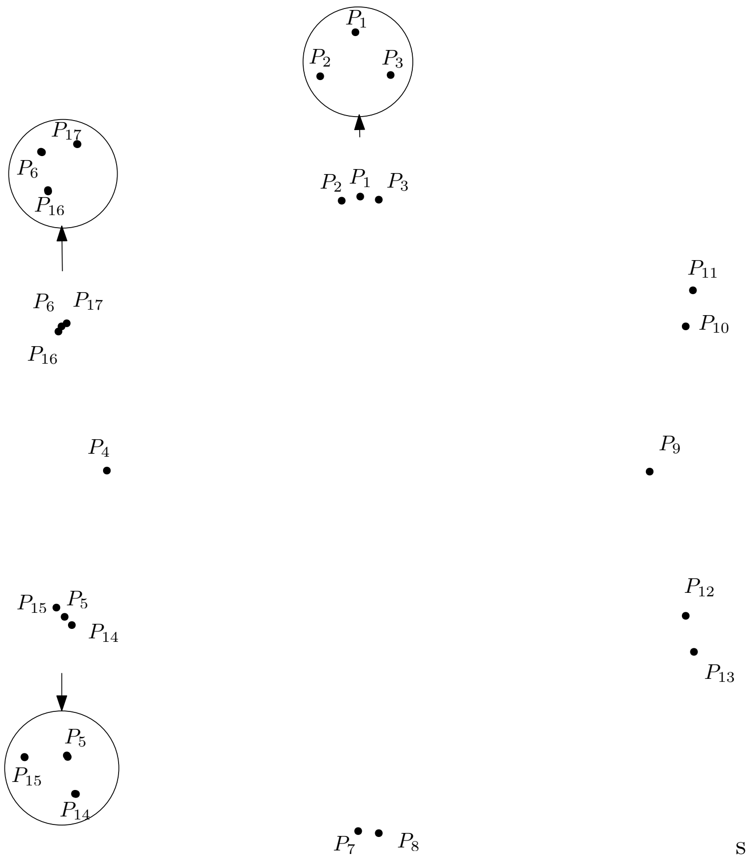


Figure 2 A set P of 17 points for Example 1.1. A zoom-in is applied to nearly linear points.

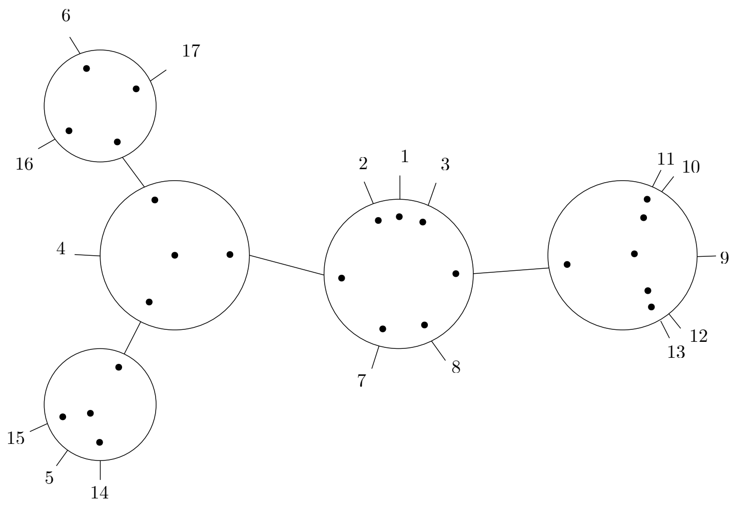


Figure 3 The tree decomposition for the points of Figure 2.

In [2, 3], this tree decomposition is built using a so-called "bowtie-operation" in an inductive way. The main result is that this yields a well-defined unique (canonical) tree having the properties alluded to above. In the present paper, rather than using this operation, we use the fact that the family of mutually avoiding bipartitions is **symmetric-crossing**. This result was proven in [3] in their construction, and we prove it again in another way.

From this, we can address and characterize the decomposition tree in terms of set families, yielding an alternative viewpoint. We use seminal results from [7], and their formulations from [5], to represent a symmetric-crossing family by a tree (its "crossing-free representation").

Then, in this way, and using an appropriate data structure (equivalent to a representation of a rank-2 uniform oriented matroid), we can efficiently compute the tree decomposition. Our method runs in time $O(n^4)$ where n is the size of the ground set E of M, and can be improved to run in time $O(n^3)$ using techniques from [10]. We thus provide an answer to the open question of [3, Paragraph 1.4 (1)]: "How efficiently can one compute the canonical chirotope tree of a given rank-3 uniform acyclic oriented matroid?".

Let us mention that various classical families can be represented by a tree in similar ways, such as those of modules of a (directed) graph, or splits of a graph (e.g. see [7, 6, 11, 9]). We refer to the thesis [4] for a state-of-the-art on representing a set family by a tree.

2 Oriented matroids, mutually avoiding sets, and tree decomposition

Oriented matroids [1] are combinatorial structures that can be used to study incidence and convexity relations in affine point configurations (but not only), and that can be defined in several equivalent ways. Here, we only need to define them by means of their chirotopes. The reader not familiar with oriented matroids does not need the following definition to understand the paper and can focus on the realizable case (of real points) presented next.

▶ **Definition 2.1.** A **rank-3 uniform oriented matroid** is defined by its **chirotope** χ, which is a mapping $\chi : E^3 \to \{-, 0, +\}$ that satisfies the following axioms:

- (Uniform) If two elements are equal amongst x_1, x_2, x_3 then $\chi(x_1, x_2, x_3) = 0$, otherwise $\chi(x_1, x_2, x_3) \neq 0$.
- (Alternating) For any permutation σ and $x_1, x_2, x_3 \in E$, $\chi(x_{\sigma(1)}, x_{\sigma(2)}, x_{\sigma(3)}) = \mathrm{sign}(\sigma) \cdot \chi(x_1, x_2, x_3)$.

- (Three terms Grassmann-Plücker relation) For all $b_1, ..., b_3, x, y \in E$, if $\chi(x, b_2, b_3).\chi(b_1, y, b_3) > 0$ and $\chi(y, b_2, b_3).\chi(x, b_1, b_3) > 0$, then $\chi(b_1, b_2, b_3).\chi(x, y, b_3) > 0$.

In this extended abstract we will not define **acyclic** nor **convex** for an oriented matroid; see [1] or [8]. For a point configuration, it is always acyclic, and convex has the usual meaning.

▶ **Definition 2.2.** Let us fix an orientation of the plane (clockwise or counterclockwise) for the entire paper. An acyclic uniform rank-3 oriented matroid M on the ground set $E = \{1, ..., n\}$ is said to be **realizable** if and only if there is an affine configuration of points in general position $P = \{p_1, ..., p_n\} \subseteq \mathbb{R}^2$ such that, for each $B = (e_1, e_2, e_3) \in E^3$, we have $\chi(B) = +$ if the triple of points $(p_{e_1}, p_{e_2}, p_{e_3})$ follows the orientation of the plane, and $\chi(B) = -$ otherwise. (The value of χ can also be defined as the sign of a determinant.) Then, P is called a **realization** of M, and we may say that M is the **oriented matroid associated to** P. (Not every uniform rank-3 oriented matroid M is realizable.)

We use the terminology of "mutually avoiding" partitions from [3], although the following definition appears as "modular bipartition" in [3, Definition 3.5]. (In fact, there are some structural similarities with modules in graphs.) In what follows, we denote $E \setminus A$ by $\overline{A}$.

▶ **Definition 2.3.** Let M be an acyclic rank-3 uniform oriented matroid on E. Let $A \subseteq E$. The pair of sets $(A, \overline{A})$ is said to be **mutually avoiding** in M if and only if for all $x_1, x_2 \in A$ and for all $y_1, y_2 \in \overline{A}$, $\chi(x_1, x_2, y_1) = \chi(x_1, x_2, y_2)$ and $\chi(y_1, y_2, x_1) = \chi(y_1, y_2, x_2)$. Abusively, we say that the set A is mutually avoiding (with $\overline{A}$). Observe that if A or $\overline{A}$ have size 0 or 1 then $(A, \overline{A})$ is considered as mutually avoiding.

Our definition below for the tree decomposition is different from the one of [3]. We can actually use two equivalent viewpoints: the one from [3] using the "bowtie-operation" and the one below. This equivalence is a reformulation of the main result [3, Theorem 1.3]. (The statement below actually contains several statements at once, but we do not detail this here.)

▶ **Definition 2.4.** Let M be a rank-3 acyclic uniform oriented matroid on E. The **(canonical) decomposition tree** $T_{\mathcal{C}}$ of M is the unique tree such that the following properties hold.

- The leaves of $T_{\mathcal{C}}$ are (in bijection with) the elements of E.
- Each (non-leaf) node x is associated to an oriented matroid $\mathcal{M}(x)$ defined on a ground set $E' \subseteq E$ chosen in the following way. Consider the subsets $E_1, ..., E_k$ of E given by leaves of the k subtrees obtained by removing x from $T_{\mathcal{C}}$ (where k is the degree of x). In every subset E_i, $1 \leq i \leq k$, we choose exactly one element, i.e., we have $|E_i \cap E'| = 1$. Elements of E' are called **representatives** for x.
- Up to isomorphism, $\mathcal{M}(x)$ does not depend on the choice of these representatives.
- For each node x, the oriented matroid $\mathcal{M}(x)$ is either convex or indecomposable (that is, with no mutually avoiding sets).
- No two adjacent nodes of $T_{\mathcal{C}}$ are associated to convex oriented matroids.
- A bipartition $(A, \overline{A})$ of E is mutually avoiding if and only if:
  - either there is an edge e of $T_{\mathcal{C}}$ such that the sets of leaves of the two subtrees obtained by removing e from $T_{\mathcal{C}}$ are A and $\overline{A}$;
  - or there is a node x such that A is a union of some subsets amongst the subsets $E_1, ..., E_k$ as defined above, and such that the restriction of $(A, \overline{A})$ to the set of representatives for x is mutually avoiding in $\mathcal{M}(x)$.

▶ **Proposition 2.5** (Rewriting of [3] Lemma 3.9). *Let $\mathcal{F}$ be the family of sets $A \subseteq E$ such that $(A, \overline{A})$ is mutually avoiding in M. Then $\mathcal{F}$ is symmetric-crossing (the definition follows).*

3 Known results about representing symmetric-crossing families

In this section we will introduce the results from [5] to represent **symmetric-crossing families** by trees. An example is shown on Figure 4. By convention, when $\mathcal{F} \subseteq 2^E$ is a family over E then $E, \emptyset$ and all the singletons of E are in $\mathcal{F}$. Two sets $A, B \subseteq E$ are said to be **crossing** if and only if $A \cap B$, $A \setminus B$, $B \setminus A$, and $\overline{A} \cap \overline{B}$ are non-empty. A set $A \in \mathcal{F}$ is said to be **crossing-free** if there is no set $B \in \mathcal{F}$ such that A and B are crossing. A family $\mathcal{F} \subseteq 2^E$ is said to be **symmetric** if, for all $A \in \mathcal{F}$, we have $\overline{A} \in \mathcal{F}$.

▶ **Definition 3.1.** A family $\mathcal{F}$ on a set E is said to be **symmetric-crossing** if it is symmetric and, for all $A, B \in \mathcal{F}$ that are crossing, we have $A \cap B \in \mathcal{F}$.

The following theorem/definition is a rewriting of a result from [7] rewritten in [5].

▶ **Theorem 3.2.** *Let $\mathcal{F} \subseteq 2^E$ be a symmetric-crossing family. Then there exists a unique* ***crossing-free representation*** *of $\mathcal{F}$ defined as follows. First, it consists of a tree $T_\mathcal{C}$ whose leaves are (in bijection with) the elements of E. Second, every (non-leaf) node has a type, either Complete, Circular, or Prime, and each Circular node is given a circular ordering of its adjacent eddges. Third, for all $A \subseteq E$, the set A belongs to $\mathcal{F}$ if and only if one of the following cases (a), (b), or (c) is true. (Note that the three cases are mutually exclusive.)*

(a) There is an edge e of $T_\mathcal{C}$ such that the sets of leaves of the two subtrees obtained by removing e from $T_\mathcal{C}$ are A and $\overline{A}$. (In this case, A and $\overline{A}$ are crossing-free in $\mathcal{F}$.)

(b) There is a node x of type Complete such that A is a union of at least 2 and at most $k-2$ subsets amongst the subsets $E_1, ..., E_k$, where k denotes the degree of x and $E_1, ..., E_k$ denote the subsets of E given by leaves of the k subtrees obtained by removing x from $T_\mathcal{C}$.

(c) There is a node x of type Circular such that A is a union of at least 2 and at most $k-2$ subsets amongst the subsets $E_1, ..., E_k$ as defined in case (b), and these subsets correspond to consecutive edges in the circular ordering given for x.

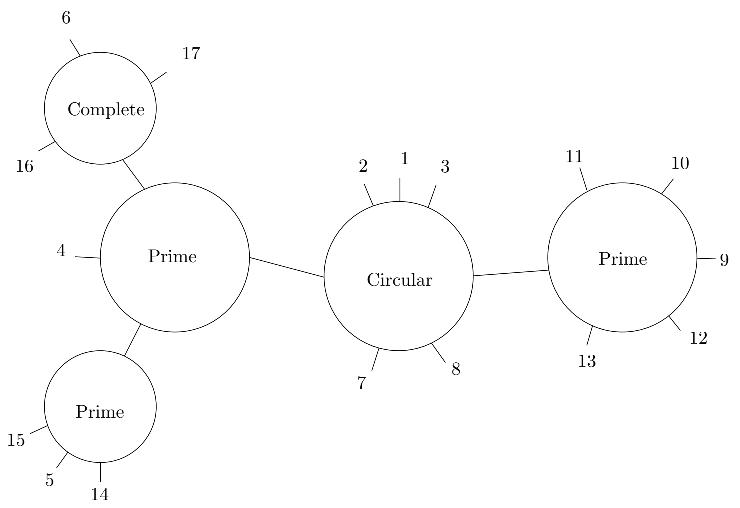


Figure 4 A crossing-free tree representation of a symmetric-crossing family $\mathcal{F}$ over $E = \{1, ..., 17\}$. The leaves of the node of type Circular are displayed around the node consistently with the circular ordering at this node. Hence, for example, by case (c) of Definition 3.2, $\{1, 2, 3, 9, 10, 11, 12, 13\}$ is a member of $\mathcal{F}$ but not $\{1, 2, 7\}$. By case (a), $\{6, 16, 17\} \in \mathcal{F}$. By case (b), $\{16, 17\} \in \mathcal{F}$.

4 The data structure used: symmetrized orderings

Let us introduce symmetrized orderings, a simple combinatorial tool that we use to construct the family of mutually avoiding bipartitions of a rank-3 uniform oriented matroid.

▶ **Definition 4.1** (Symmetrized ordering)**.** Let X be a ground set. Consider a set of diameters of a circle, one for each $e \in X$, such that one of the two endpoints is labeled by e and the other one by $\overline{e}$. Then, the circular ordering of these labels along the circle (following the orientation of the plane) is called a **symmetrized ordering** (over X).

The outer circle in Figure 5 shows an example of a symmetrized ordering. Symmetrized orderings trivially correspond to representations of uniform rank-2 oriented matroids (except that, in oriented matroids, replacing all the elements by their opposites yields the same oriented matroid). As a consequence, if M is a uniform rank-3 oriented matroid on the ground set E, then, for any $e \in E$, the rank-2 oriented matroid M/e can be represented by a symmetrized ordering σ_e. See [1] for the general definition of M/e. In the realizable case, M/e is obtained by projecting all elements of $E \setminus e$ onto a circle centered at e. The example of a symmetrized ordering in Figure 5 is obtained in this way in the realizable case.

An **interval** of a symmetrized ordering σ over a set E is a subset A of element of σ such that the elements of A appear consecutively in σ (in any order) and all the element of A are in E (and not opposites of elements of E).

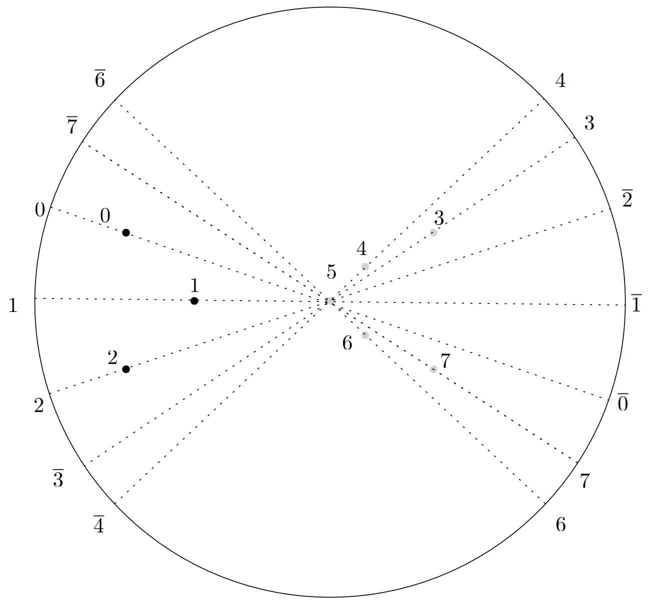


Figure 5 A symmetrized ordering obtained from the set of points of Figure 2 associated to $M/5$. In clockwise ordering, we obtain $\sigma_5 = 4, 3, \overline{2}, \overline{1}, \overline{0}, 7, 6, \overline{4}, \overline{3}, 2, 1, 0, \overline{7}, \overline{6}$. The set $\{0, 1, 2\}$ is an interval, but not $\{1, 2, 3, 4\}$ and neither $\{\overline{1}, \overline{2}, 3, 4\}$.

5 Contributions

In all the statements below, M is a rank-3 acyclic uniform oriented matroid on $E = \{1, ..., n\}$, and $\mathcal{F}$ is the family of mutually avoidings sets of M.

▶ **Theorem 5.1.** *Let $A \subseteq E$. Then, $A \in \mathcal{F}$ if and only if, for every $e \in E$, either A or $\overline{A}$ is an interval in the symmetrized ordering σ_e associated to M/e.*

Sketch of proof. Geometrically, this reformulates mutual avoidance as a local property around each vertex, based on lines formed with other vertices. See details in [8]. ◀

As a corollary, one obtains a new proof of Proposition 2.5, since the family $\mathcal{F}$ can be viewed as an intersection of symmetric-crossing families. Then, the next theorem gives a new perspective on the decomposition tree, by expressing it as a crossing-free tree representation. Figures 3 and 4 show these two related tree representations for the example of Figure 2.

▶ **Theorem 5.2.** *The canonical tree decomposition of M is obtained by decorating the nodes of the crossing-free representation of $\mathcal{F}$ (in particular, the underlying trees are the same).*

Sketch of proof. Since $\mathcal{F}$ is symmetric-crossing by Proposition 2.5, $\mathcal{F}$ has a crossing-free representation with tree $T_{\mathcal{C}}$ by Theorem 3.2. One has to relate this setting to Definition 2.4.

- In the tree $T_{\mathcal{C}}$, the leaves are in bijection with E.
- Let x be a node of this tree and let $E_1, ..., E_k$ be the subsets of leaves given by the subtrees $T_1, ..., T_k$ obtained by removing x. To each node x we associate an oriented matroid $\mathcal{M}(x)$ defined on a set E' such that $E' \cap E_i = e_i$ for all $1 \leq i \leq k$, and we show that the choice of the representative elements $e_i \in E_i$ does not affect the resulting $\mathcal{M}(x)$.
- We show that x cannot be of type Complete. Furthermore, we show that if x is of type *Circular* or *Prime* then $\mathcal{M}(x)$ is convex or indecomposable, respectively.
- We show that two nodes that are adjacent in $T_{\mathcal{C}}$ cannot both be decorated by convex oriented matroids.
- A subset A of E is in $\mathcal{F}$ if and only if there is a node x such that A is a union of the E_i's of x and the restriction of $(A, \overline{A})$ to E' is mutually avoiding in $M(E')$.

With those claims, we have shown that $T_{\mathcal{C}}$ decorated by the oriented matroids $\mathcal{M}(x)$ exactly corresponds to the canonical tree decomposition of Definition 2.4. ◀

▶ **Theorem 5.3.** *The family $\mathcal{F}$ can be computed in time $O(n^3)$ and has size $O(n^2)$. It implies that we can compute the canonical tree decomposition of M in time $O(n^4)$.*

Sketch of proof. Since the number of intervals of any σ_e is $O(n^2)$, $\mathcal{F}$ has size $O(n^2)$. We can show that computing any σ_e can be done in time $O(n \cdot \log(n))$ hence the set of intervals of any σ_e can be computed in time $O(n^2)$ which implies that we can compute $\mathcal{F}$ in time $O(n^3)$. We can compute the crossing-free representation of $\mathcal{F}$ as follows. First, we compute the set of crossing-free elements of $\mathcal{F}$. This can be done in time $O(n^4)$ since $\mathcal{F}$ has size $O(n^2)$. Then, we sort those sets by inclusion to obtain the shape of the tree. This can be done in time $O(n^2)$ since there is $O(n)$ crossing-free elements. Lastly, for every node x, we can test if its associated oriented matroid is convex in time $O(n^3)$. If it is, then we can find an associated circular ordering in time $O(n \cdot \log(n))$. By marking every node by their type or decorating it by its corresponding oriented matroid, we obtain either the crossing-free representation of $\mathcal{F}$ or the canonical tree decomposition of M. The total computation time is $O(n^4)$. ◀

The above method is fairly easy to compute. The computation time can be improved using [10] (precisely Theorem 2, Corollary 1, Property 4, and part of the proof of Theorem 8).

▶ **Theorem 5.4.** *We can compute the canonical tree decomposition of M in time $O(n^3)$.*

Acknowledgments. We wish to thank Christophe Paul and Hugo Jacob for useful discussions.

References

1 Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Gunter M. Ziegler. *Oriented Matroids*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2 edition, 1999.

2 Mathilde Bouvel, Valentin Féray, Xavier Goaoc, and Florent Koechlin. A Canonical Tree Decomposition for Chirotopes. In *40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:18, 2024.

3 Mathilde Bouvel, Valentin Féray, Xavier Goaoc, and Florent Koechlin. A canonical tree decomposition for order types, and some applications. arXiv:2403.10311, 2024.

4 Binh-Minh Bui-Xuan. *Tree-Representation of Set Families in Graph Decompositions and Efficient Algorithms*. Theses, Université Montpellier II - Sciences et Techniques du Languedoc, September 2008.

5 Binh-Minh Bui-Xuan, Michel Habib, and Michaël Rao. Tree-representation of set families and applications to combinatorial decompositions. *European Journal of Combinatorics*, 33(5):688–711, 2012. EuroComb '09.

6 William H. Cunningham. Decomposition of directed graphs. *SIAM Journal on Algebraic Discrete Methods*, 3:214–228, 1982.

7 William H. Cunningham and Jack Edmonds. A combinatorial decomposition theory. *Canadian Journal of Mathematics*, 32:734 – 765, 1980.

8 Emeric Gioan and Yann Marin. Computation of the canonical tree decomposition of a rank-3 uniform oriented matroid into mutually avoiding parts. Full-length paper of this extended abstract. In preparation. A repository for preliminary versions is available here:: `https://hal-lirmm.ccsd.cnrs.fr/lirmm-04905101`, 2025.

9 Emeric Gioan and Christophe Paul. Split decomposition and graph-labelled trees: Characterizations and fully dynamic algorithms for totally decomposable graphs. *Discrete Applied Mathematics*, 160(6):708–733, 2012.

10 Michel Habib, Fabien Montgolfier, Lalla Mouatadid, and Mengchuan Zou. A general algorithmic scheme for combinatorial decompositions with application to modular decompositions of hypergraphs. *Theoretical Computer Science*, 923:56–73, 2022.

11 Michel Habib and Christophe Paul. A survey on algorithmic aspects of modular decomposition. *Computer Science Review*, 4:41–59, 2010.

Valleys for Continuous Dynamic Time Warping

Maike Buchin[1] **and Jan Erik Swiadek**[1]

1 **Faculty of Computer Science, Ruhr-Universität Bochum**
`{maike.buchin,jan.swiadek}@rub.de`

Abstract

Valleys are objects central to computing Continuous Dynamic Time Warping for polygonal curves. We study how the choice of the underlying distance function impacts existence and properties of valleys. In particular, we show that not all metrics guarantee their existence but that all seminorm-induced pseudometrics do. The metric that is induced by the vector space's equipped norm even admits a constructive characterisation of valleys especially suited for use in CDTW algorithms.

1 Introduction and Preliminaries

Continuous Dynamic Time Warping (CDTW) has been employed in approaches to measuring the similarity of polygonal curves robustly to outliers and sampling rates [4, 5]. A polygonal curve P in a normed vector space $(\mathbb{R}^2, \|\cdot\|)$ is represented by a point sequence $\langle p_0, \dots, p_n \rangle$ with $n \in \mathbb{N}$ and $p_i \neq p_{i-1}$ for $i \in \{1, \dots, n\}$. Its arc length is $\|P\| := \sum_{i=1}^{n} \|p_i - p_{i-1}\|$ and we denote its arc length parametrisation by $P_{\|\cdot\|} \colon [0, \|P\|] \to \mathbb{R}^2$. Given polygonal curves P, Q, the considered CDTW definition under norm $\|\cdot\|$ and distance function $d \colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$ goes back to the work in [6, Chapter 6] and can be stated as follows [5, Lemma 1]:

$$\mathrm{cdtw}_{\|\cdot\|,d}(P, Q) := \inf_{\gamma \in \Psi_{\|\cdot\|}(P,Q)} \int_0^{\|P\|+\|Q\|} d(P_{\|\cdot\|}(\gamma_1(z)), Q_{\|\cdot\|}(\gamma_2(z)))\, \mathrm{d}z.$$

The set $\Psi_{\|\cdot\|}(P, Q)$ contains monotone paths $\gamma \colon [0, \|P\| + \|Q\|] \to [0, \|P\|] \times [0, \|Q\|]$ through the joint parameter space of P, Q such that there are piecewise continuously differentiable parametrisations $f_P, f_Q \colon [0, \|P\|+\|Q\|] \to \mathbb{R}^2$ of P, Q respectively with $\gamma_1(z) = \int_0^z \|f_P'(y)\|\, \mathrm{d}y$ and $\gamma_2(z) = \int_0^z \|f_Q'(y)\|\, \mathrm{d}y$ as well as $\gamma_1'(z) + \gamma_2'(z) = 1$ for all $z \in [0, \|P\| + \|Q\|]$.

Computing CDTW is therefore equivalent to finding an optimal monotone path through this parameter space, similar to other continuous measures such as the Fréchet distance [1]. It is in fact closely related to computing a lexicographically optimal Fréchet path [11], which minimises a profile function of attained distance values lexicographically. The profile measures for how long every distance value is exceeded, so CDTW minimises the area under the profile. Optimal CDTW paths depend on the norm $\|\cdot\|$ that scales the parameter space, the distance function d that endows the parameter space with a terrain, and the regularisation constraint that limits the speed of paths. The above definition regularises the sum of component speeds to equal 1, so that $\gamma(z) = (x_1, x_2)^\intercal$ for any path $\gamma \in \Psi_{\|\cdot\|}(P, Q)$ implies $x_1 + x_2 = z$.

Intuitively, the points in parameter space reachable after duration exactly z are the ones on the line of slope -1 through $\gamma(z)$. Since CDTW realises steepest monotone descent in the terrain [11], optimal subpaths for a pair of curve segments may be induced by minima of d on lines of slope -1. Exact CDTW algorithms utilise this by dividing the parameter space into cells that correspond to segment pairs, computing costs of optimal subpaths within each cell, and propagating optimum cost functions through the grid of cells in a dynamic program [5, 3]. This requires that optimal subpaths and optimum cost functions are well-behaved. We are interested in the former and thus focus on *polygonal segments*, i.e. curves P, Q with $n = 1$ segment each, while extending the domains of their arc length parametrisations to $\mathbb{R}$.

Minima relevant for optimal paths are then attained on a line, dubbed *valley* [5], under previously used distance functions, namely the 2-metric [10] and the 1-metric [3, Chapter 5]. We study existence and properties of valleys more generally, motivated by the aforementioned optimum cost functions being unwieldy under e.g. the 2-metric. Exact CDTW computations seem prone to algebraic issues in such cases [5, 9]. Hence, admitting a large class of distance functions can enable approximations of a desired function through more well-behaved ones. To ensure that the (Lebesgue) integral from the CDTW definition exists as a value in $[0, \infty]$, we consider lower semicontinuous functions, i.e. functions whose sublevel sets are closed in the topology given by the equipped norm $\|\cdot\|$. For a real number $\mu \in \mathbb{R}$ the *μ-sublevel set* of a real-valued function $f\colon \mathrm{dom}(f) \to \mathbb{R}$ is defined as $S_{\le\mu}(f) := \{x \in \mathrm{dom}(f) \mid f(x) \le \mu\}$.

▶ **Definition 1.1** ([3, Definition 2]). Let $d\colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}_{\ge 0}$ be lower semicontinuous. A *valley* under d for polygonal segments P, Q is a line $\ell \subseteq \mathbb{R}^2$ not of slope -1 such that for almost every $(x_1, x_2)^\mathsf{T} \in \ell$ the function $y \mapsto d(P_{\|\cdot\|}(x_1 + y), Q_{\|\cdot\|}(x_2 - y))$ is non-increasing on $\mathbb{R}_{\le 0}$ and non-decreasing on $\mathbb{R}_{\ge 0}$. If all P, Q have a valley under d, then d *guarantees valleys.*

In the following, we say for short that a point $(x_1, x_2)^\mathsf{T} \in \mathbb{R}^2$ with the above property has the *monotonicity property.* It suffices that almost every point of a valley ℓ has this property, i.e. we may exclude a null subset of ℓ, because this has no effect on the CDTW integral.

Furthermore, we make use of gauge functions given by their 1-sublevel sets: Let $K \subseteq \mathbb{R}^2$ be absorbing and balanced (i.e. both $\bigcup\{\lambda K \mid \lambda \in \mathbb{R}\} = \mathbb{R}^2$ and $\bigcup\{\lambda K \mid \lambda \in [-1, 1]\} \subseteq K$ hold) as well as closed. The *gauge* of K is $\mathcal{G}_K\colon \mathbb{R}^2 \to \mathbb{R}_{\ge 0}$ with $\mathcal{G}_K(x) := \inf\{\lambda \in \mathbb{R}_{\ge 0} \mid x \in \lambda K\}$, which satisfies $S_{\le\mu}(\mathcal{G}_K) = \mu K$ for all $\mu \in \mathbb{R}_{\ge 0}$ by the assumptions on K. If K is moreover convex (i.e. $\lambda x + (1 - \lambda)x' \in K$ holds for all pairs $x, x' \in K$ and all $\lambda \in [0, 1]$), then $\mathcal{G}_K$ is a seminorm and induces a pseudometric, and if K is additionally bounded with regard to $\|\cdot\|$, then $\mathcal{G}_K$ is a norm and induces a metric. Conversely, every norm or seminorm is the gauge of its absorbing, balanced, convex and closed 1-sublevel set. (See [12, pp. 39–40].) We define sets $M(p) := \{(x_1, x_2)^\mathsf{T} \in \mathbb{R}^2 \mid (|x_1|^p + |x_2|^p)^{1/p} \le 1\}$, so that $\mathcal{G}_{M(p)}$ is the p-norm for $p \ge 1$ and thus induces the p-metric $(p', q') \mapsto \mathcal{G}_{M(p)}(p' - q')$ on $\mathbb{R}^2$. For $p \in (0, 1)$ the gauge $\mathcal{G}_{M(p)}$ is not a norm, but in this case the function $(p', q') \mapsto [\mathcal{G}_{M(p)}(p' - q')]^p$ is a metric on $\mathbb{R}^2$.

2 Basic Insights and Examples

The monotonicity property for valleys from Definition 1.1 is based on quasiconvex functions, i.e. functions whose sublevel sets are convex (cf. [2, Section 3.4]). As quasiconvexity is invariant to compositions with non-decreasing functions, this also applies to distance functions that guarantee valleys. Furthermore, every such distance function d satisfies a necessary condition that is related to joint quasiconvexity. For continuous d it even implies joint quasiconvexity if d is also translation-invariant, i.e. if $d(p, q) = d(p + t, q + t)$ holds for all $p, q, t \in \mathbb{R}^2$.

▶ **Theorem 2.1.** *Let $d\colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}_{\ge 0}$ be lower semicontinuous and guarantee valleys.*

1. *If $f\colon \mathbb{R}_{\ge 0} \to \mathbb{R}_{\ge 0}$ is lower semicontinuous and non-decreasing, $f \circ d$ guarantees valleys.*
2. *Provided d is continuous at $(p, q), (p', q') \in S_{\le\mu}(d)$ with $\|p' - p\| = \|q' - q\|$ and $\mu \in \mathbb{R}_{\ge 0}$, the μ-sublevel set $S_{\le\mu}(d)$ of d contains $(\lambda p + (1 - \lambda)p', \lambda q + (1 - \lambda)q')$ for all $\lambda \in [0, 1]$.*
3. *In case d is continuous everywhere and translation-invariant, d is jointly quasiconvex.*

Proof. 1. Because f is non-decreasing, we have $d(p, q) \le d(p', q') \Leftrightarrow f(d(p, q)) \le f(d(p', q'))$ for $(p, q), (p', q') \in \mathbb{R}^2 \times \mathbb{R}^2$. A valley under d for polygonal segments P, Q thus is a valley under $f \circ d$ too, as f cannot invalidate the monotonicity property from Definition 1.1.

2. Consider polygonal segments $P := \langle p, p' \rangle$ and $Q := \langle q', q \rangle$. It is $(P_{\|\cdot\|}(0), Q_{\|\cdot\|}(z_0)) = (p, q)$ and $(P_{\|\cdot\|}(z_0), Q_{\|\cdot\|}(0)) = (p', q')$ with $z_0 := \|p' - p\| = \|q' - q\|$. Further let ℓ be a valley under d for P, Q and let $(x_1, x_2)^\top \in \ell$ with $x_1 + x_2 = z_0$. If the monotonicity property from Definition 1.1 holds for $(x_1, x_2)^\top$, the values attained by d between (p, q) and (p', q') are either at first non-increasing and then non-decreasing or only one of the two, which gives $d(\lambda p + (1 - \lambda)p', \lambda q + (1 - \lambda)q') \leq \max\{d(p, q), d(p', q')\} \leq \mu$ for all $\lambda \in [0, 1]$.
   Otherwise, the continuity of d at $(p, q), (p', q')$ implies that for each $\varepsilon > 0$ there is a $\delta_\varepsilon > 0$ with $d(P_{\|\cdot\|}(\delta), q) \leq \mu + \varepsilon$ and $d(P_{\|\cdot\|}(z_0 + \delta), q') \leq \mu + \varepsilon$ for all $\delta \in [-\delta_\varepsilon, \delta_\varepsilon]$. Also, the monotonicity property holds for almost every $(x_1', x_2')^\top \in \ell$ with $x_1' + x_2' - z_0 \in [-\delta_\varepsilon, \delta_\varepsilon]$, so all points between (p, q) and (p', q') are limit points of $S_{\leq \mu+\varepsilon}(d)$. They are therefore contained in $\bigcap\{S_{\leq \mu+\varepsilon}(d) \mid \varepsilon > 0\} = S_{\leq \mu}(d)$ since the sublevel sets of d are closed.
3. Let $(p, q), (p', q') \in S_{\leq \mu}(d)$ with $\mu \in \mathbb{R}_{\geq 0}$, and let $x := p - q$, $x' := p' - q'$, $t := (x' - x)/2$. The translation-invariance of d gives $d(x', \mathbf{0}) = d(p', q')$ and $d(x + t, t) = d(p, q)$. Now (2) applies to $(x + t, t), (x', \mathbf{0}) \in S_{\leq \mu}(d)$ due to $\|x' - (x + t)\| = \|t\| = \|\mathbf{0} - t\|$ and continuity of d, so $d(\lambda(x + t) + (1 - \lambda)x', \lambda t) \leq \mu$ holds for all $\lambda \in [0, 1]$. Using translation-invariance again yields $(\lambda p + (1 - \lambda)p', \lambda q + (1 - \lambda)q') \in S_{\leq \mu}(d)$ for all $\lambda \in [0, 1]$ as desired. ◀

▶ **Example 2.2.** The 2-metric guarantees valleys of slope 1 if $\|\cdot\|$ is the 2-norm [10] since the terrain in parameter space then consists of ellipses that have axes of slope 1 and -1 [11], see Figure 1a for $P := \langle (1, 0)^\top, (-23, 7)^\top \rangle$ and $Q := \langle (0, 0)^\top, (0, 25)^\top \rangle$. Its non-metric square $(p, q) \mapsto [\mathcal{G}_{M(2)}(p - q)]^2$ works too by Theorem 2.1 (1) and has already been used [4].

The translation-invariant metric $(p, q) \mapsto [\mathcal{G}_{M(1/2)}(p - q)]^{1/2}$ does not guarantee valleys because it violates the condition from Theorem 2.1 (3), see Figure 1b. Meanwhile, the metric $(p, q) \mapsto \mathcal{G}_{M(1)}(p) + \mathcal{G}_{M(1)}(q)$ for $p \neq q$ and $(p, p) \mapsto 0$, which is an absolutely homogeneous post office metric, satisfies the condition from Theorem 2.1 (2) but does not guarantee valleys: Relevant minima for P, Q are not attained on a single line, as indicated in Figure 1c.

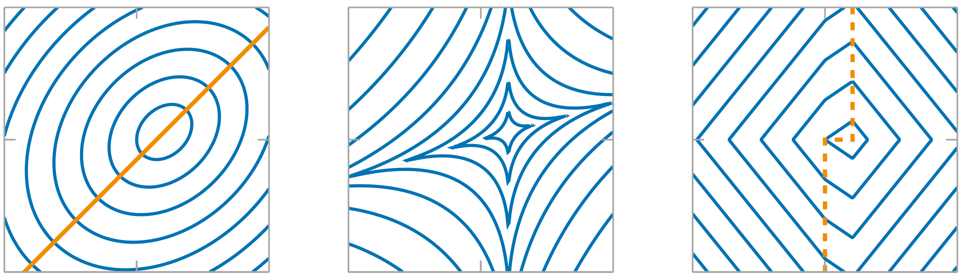

(a) 2-metric induced by 2-norm **(b)** Metric based on star set **(c)** PO metric based on 1-norm

Figure 1 Terrains under some distance functions in a parameter space with $\|\cdot\| = \mathcal{G}_{M(2)}$

Intuitively, a distance function d is rather straightforward if it is translation-invariant because then all information is given by $x \mapsto d(x, \mathbf{0})$. In Figure 1a and 1b we observe that the terrain consists of affinely transformed versions of $M(2)$ and $M(1/2)$ respectively.

▶ **Lemma 2.3** ([1, Lemma 3])**.** *Let $d \colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$ be translation-invariant and let P, Q be polygonal segments. Define $\varphi \colon \mathbb{R}^2 \to \mathbb{R}^2, \Delta \colon \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$ by $\varphi(x_1, x_2) := P_{\|\cdot\|}(x_1) - Q_{\|\cdot\|}(x_2)$ and $\Delta(x_1, x_2) := d(P_{\|\cdot\|}(x_1), Q_{\|\cdot\|}(x_2))$. If P, Q are not parallel, then the affine map φ has an affine inverse $\varphi^{-1} \colon \mathbb{R}^2 \to \mathbb{R}^2$ with $S_{\leq \mu}(\Delta) = \varphi^{-1}(S_{\leq \mu}(x \mapsto d(x, \mathbf{0})))$ for all $\mu \in \mathbb{R}_{\geq 0}$. Else, both φ and Δ are constant either on every line of slope 1 or on every line of slope -1.*

The condition from Theorem 2.1 (3) is achievable via gauges of absorbing, convex sets. A natural way to always obtain valleys is making the sets balanced too, giving seminorms. This result generalises those for the 2-metric [10, Lemma 4] and 1-metric [3, Lemma 24].

▶ **Theorem 2.4.** *Every seminorm-induced pseudometric d on $\mathbb{R}^2$ guarantees valleys.*

Proof. There is a balanced, convex and closed $K \subseteq \mathbb{R}^2$ with $d(p,q) = \mathcal{G}_K(p-q)$ for $p,q \in \mathbb{R}^2$. Let P,Q be polygonal segments and let $\varphi \colon \mathbb{R}^2 \to \mathbb{R}^2$ be as in Lemma 2.3, so for $(x_1,x_2)^\intercal \in \mathbb{R}^2$ it is $d(P_{\|\cdot\|}(x_1), Q_{\|\cdot\|}(x_2)) = \mathcal{G}_K(\varphi(x_1,x_2))$. Assume first that P,Q are parallel. Then $\mathcal{G}_K \circ \varphi$ is constant either on every line of slope 1 or -1 by Lemma 2.3. In the latter case every line ℓ not of slope -1 is a valley for P,Q by Definition 1.1. In the former case it suffices to discern one point that has the monotonicity property, so consider $L := \{\varphi(y,-y) \mid y \in \mathbb{R}\}$.

For each $\mu \in \mathbb{R}_{\geq 0}$ we have $S_{\leq\mu}(\mathcal{G}_K) = \mu K$ and $L \cap \mu K$ is either empty or the line L or a bounded segment/point on L due to the properties of K. If $L \cap \mu K \in \{\varnothing, L\}$ for all $\mu \in \mathbb{R}_{\geq 0}$, then $\mathcal{G}_K$ is constant on L and every $(y,-y)^\intercal$ with $y \in \mathbb{R}$ has the monotonicity property. Else, there is a $\mu \in \mathbb{R}_{\geq 0}$ with $L \cap \mu K$ non-empty and bounded, so $\mathcal{G}_K$ attains its minimum on L by continuity at $\varphi(y_0,-y_0)$ for a $y_0 \in \mathbb{R}$. Together with $\mathcal{G}_K$ being quasiconvex on L, as $L \cap \mu K$ is convex for all $\mu \in \mathbb{R}_{\geq 0}$, it follows that the monotonicity property from Definition 1.1 holds for $(y_0,-y_0)^\intercal$ and $\ell := \{(y_0+\lambda, -y_0+\lambda)^\intercal \mid \lambda \in \mathbb{R}\}$ is a valley of slope 1 for P,Q.

Now assume that P,Q are not parallel. Lemma 2.3 says $S_{\leq\mu}(\mathcal{G}_K \circ \varphi) = \varphi^{-1}(S_{\leq\mu}(\mathcal{G}_K)) = \varphi^{-1}(\mu K)$ for all $\mu \in \mathbb{R}_{\geq 0}$. Since φ^{-1} is a bijective affine map, the set $K' := \varphi^{-1}(K) - \varphi^{-1}(\mathbf{0})$ inherits the properties of K and further satisfies $\mu K' = \varphi^{-1}(\mu K) - \varphi^{-1}(\mathbf{0})$ for all $\mu \in \mathbb{R}_{\geq 0}$, so $\mathcal{G}_{K'} = (\mathcal{G}_K \circ \varphi) \circ (x \mapsto x + \varphi^{-1}(\mathbf{0}))$. Given any line $L_z := \{(x_1,x_2)^\intercal \in \mathbb{R}^2 \mid x_1 + x_2 = z\}$ of slope -1 with $z \in \mathbb{R}$, examining $L_z \cap \mu K'$ for $\mu \in \mathbb{R}_{\geq 0}$ works as above and yields $x_0 \in L_z$ such that the monotonicity property from Definition 1.1 holds for $x_0 + \varphi^{-1}(\mathbf{0})$.

It remains to show that there is a valley ℓ containing such points. Each line of slope -1 except L_0 is of the form $\lambda L_1 = L_\lambda$ for a $\lambda \in \mathbb{R} \setminus \{0\}$, so $z = 1$ gives $\lambda x_0 \in L_\lambda$ for all $\lambda \in \mathbb{R}$. Furthermore, we have $\mathcal{G}_{K'}(\lambda x) = |\lambda| \mathcal{G}_{K'}(x)$ for all $x \in L_1$ and all $\lambda \in \mathbb{R}$ since K' is balanced. Thus, every point in $\ell := \{\lambda x_0 + \varphi^{-1}(\mathbf{0}) \mid \lambda \in \mathbb{R}\}$ has the monotonicity property, similar to the proof of Theorem 2.1 (1) applied to $f_\lambda(y) := |\lambda| y$ for $\lambda \in \mathbb{R} \setminus \{0\}$ and $y \in \mathbb{R}_{\geq 0}$. ◀

▶ **Example 2.5.** The 1-metric guarantees valleys that can have negative slope if $\|\cdot\|$ is the 2-norm [3], see Figure 2a for $P := \langle(0.2, 0.1)^\intercal, (-19, 5.7)^\intercal\rangle$ and $Q := \langle(0.1, 0.2)^\intercal, (13, -17)^\intercal\rangle$. By Theorem 2.4 the pseudometric $(p,q) \mapsto \mathcal{G}_{[-1,1]\times\mathbb{R}}(p-q)$ works too, see Figure 2b, but note that being seminorm-induced or translation-invariant is not necessary: The post office metric $(p,q) \mapsto \mathcal{G}_{M(2)}(p) + \mathcal{G}_{M(2)}(q)$ for $p \neq q$ and $(p,p) \mapsto 0$ guarantees valleys, see Figure 2c.

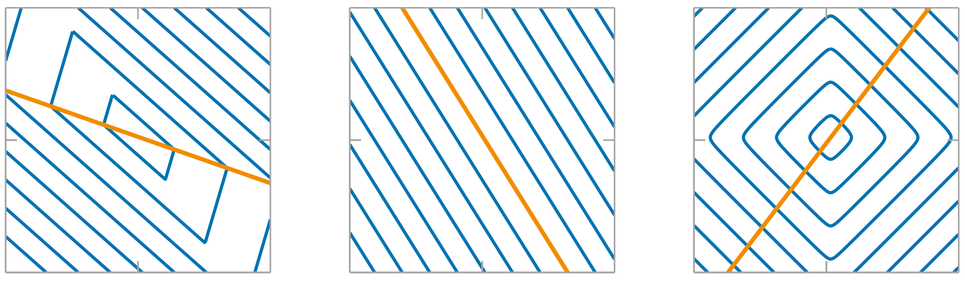

(a) 1-metric induced by 1-norm **(b)** Pseudometric based on slab **(c)** PO metric based on 2-norm

Figure 2 Terrains under more distance functions in a parameter space with $\|\cdot\| = \mathcal{G}_{M(2)}$

3 Valleys under the Equipped Norm

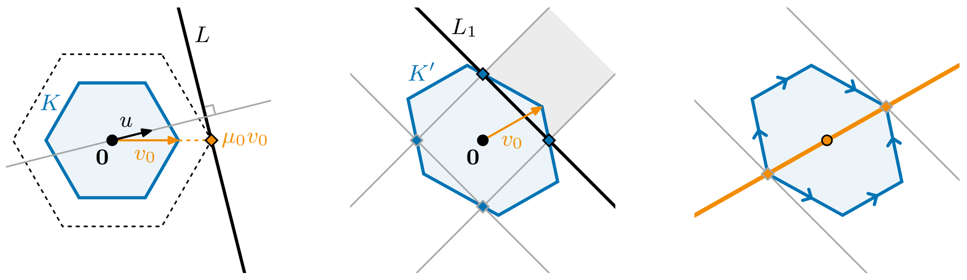
Valleys of negative slope are not so easy to deal with in CDTW algorithms since monotone paths cannot travel along them [3, Section 5.5]. We show that the metric induced by the equipped norm $\|\cdot\|$ fortunately guarantees valleys of positive slope. To this end, we identify the valley-relevant points from the proof of Theorem 2.4 that minimise a gauge on a line. The next lemma utilises duality (cf. [2, Section 5.1.6]), but we give an elementary proof.

▶ **Lemma 3.1.** *Let $L = \{x \in \mathbb{R}^2 \mid u^\intercal \cdot x = z\}$ be a line specified by $u \in \mathbb{R}^2 \setminus \{\mathbf{0}\}$ and $z \in \mathbb{R}$. The gauge $\mathcal{G}_K \colon \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$ of an absorbing, balanced, closed and bounded set $K \subseteq \mathbb{R}^2$ attains its minimum on L at $(z/z_0)v_0$, where $v_0 \in \arg\max\{u^\intercal \cdot v \mid v \in K\}$ and $z_0 := u^\intercal \cdot v_0$.*

Proof. First of all, we have $(z/z_0)v_0 \in L$ because $u^\intercal \cdot (z/z_0)v_0 = z$ holds by definition of z_0. If $z = 0$, then $(z/z_0)v_0 = \mathbf{0}$ minimises the gauge $\mathcal{G}_K$. Otherwise, for all $x \in L$ it is $x \neq \mathbf{0}$ and thus $\mathcal{G}_K(x) > 0$, as K is bounded, as well as $x/\mathcal{G}_K(x) \in K$, as K is closed. This yields

$$\mathcal{G}_K(x) = \frac{|z|}{|z|} \cdot \mathcal{G}_K(x) = \frac{|z|}{|u^\intercal \cdot x|} \cdot \mathcal{G}_K(x) = \frac{|z|}{|u^\intercal \cdot (x/\mathcal{G}_K(x))|} \geq \frac{|z|}{|u^\intercal \cdot v_0|} = \frac{|z|}{|z_0|}$$

for all $x \in L$, where the inequality follows from v_0 maximising $v \mapsto u^\intercal \cdot v$ on K and thereby also maximising $v \mapsto |u^\intercal \cdot v|$ on K, as K is balanced. Furthermore, $v_0 \in K$ and K being balanced also imply $|z/z_0| \geq |z/z_0|\mathcal{G}_K(v_0) = \mathcal{G}_K((z/z_0)v_0)$, which completes the proof. ◀



(a) Minimising norm on a line **(b)** Identifying positive slope **(c)** Computing valley points

Figure 3 Characterisation of valleys under a norm with regular hexagons as sublevel sets

Geometrically, Lemma 3.1 can be interpreted as follows: Scale K by $\mu_0 = |z/z_0| \in \mathbb{R}_{\geq 0}$ such that L is a supporting line of $\mu_0 K$, i.e. $L \cap \mu_0 K \neq \varnothing$ and $\mu_0 K$ is contained in a closed half-plane given by L. The minimum of $\mathcal{G}_K$ on L is attained on $L \cap \mu_0 K$, see Figure 3a.

▶ **Theorem 3.2.** *The metric d induced by the equipped norm $\|\cdot\|$ on $\mathbb{R}^2$ guarantees valleys of positive slope. Computing such a valley for polygonal segments $P = \langle p_0, p_1\rangle$ and $Q = \langle q_0, q_1\rangle$ reduces to Lemma 3.1 with $K := S_{\leq 1}(\|\cdot\|)$ and $u \perp (p_1 - p_0)$ if P, Q are parallel, or else with $K' := \varphi^{-1}(K) - \varphi^{-1}(\mathbf{0})$ and $u' := (1,1)^\intercal$, where $\varphi \colon \mathbb{R}^2 \to \mathbb{R}^2$ is as in Lemma 2.3.*

Proof. It is $d(P_{\|\cdot\|}(x_1), Q_{\|\cdot\|}(x_2)) = \|\varphi(x_1, x_2)\| = \mathcal{G}_K(\varphi(x_1, x_2))$ for $(x_1, x_2)^\intercal \in \mathbb{R}^2$. If P, Q are parallel, the proof of Theorem 2.4 says that they have a valley of positive slope under d: In case of opposite directions, every line not of slope -1 is a valley, while the codirectional case yields a valley of slope 1 through $(y_0, -y_0)^\intercal \in \mathbb{R}^2$ such that $\mathcal{G}_K$ attains its minimum on $L := \{\varphi(y, -y) \mid y \in \mathbb{R}\}$ at $\varphi(y_0, -y_0)$. It then is $\varphi(y, -y) = 2y \cdot \frac{p_1 - p_0}{\|p_1 - p_0\|} + \varphi(\mathbf{0})$ for $y \in \mathbb{R}$, so Lemma 3.1 applies using a $u \in \mathbb{R}^2 \setminus \{\mathbf{0}\}$ with $u^\intercal \cdot (p_1 - p_0) = 0$ and $z := u^\intercal \cdot \varphi(\mathbf{0})$.

If P, Q are not parallel, the proof of Theorem 2.4 says that there is a valley through $\varphi^{-1}(\mathbf{0})$ and $x_0 + \varphi^{-1}(\mathbf{0})$ such that $x_0 \in L_1$ minimises $\mathcal{G}_{K'}$ on $L_1 := \{(x_1, x_2)^\intercal \in \mathbb{R}^2 \mid x_1 + x_2 = 1\}$. It is $x_0 = (1/z_0)v_0$ with $v_0 \in \arg\max\{(1,1) \cdot v \mid v \in K'\}$ and $z_0 := (1,1) \cdot v_0$ by Lemma 3.1, meaning the goal is to find such a v_0 with positive components. The affine map φ gives unit vectors $\varphi(\pm 1, 0) - \varphi(\mathbf{0}) = \pm\frac{p_1 - p_0}{\|p_1 - p_0\|}$ and $\varphi(0, \pm 1) - \varphi(\mathbf{0}) = \mp\frac{q_1 - q_0}{\|q_1 - q_0\|}$ in the boundary of K, so the boundary of K' contains $(\pm 1, 0)^\intercal$ and $(0, \pm 1)^\intercal$. As K' is closed and convex, we have $L_1 \cap \mathbb{R}^2_{\geq 0} \subseteq \{v \in K' \mid (1,1) \cdot v \geq 1\} \subseteq L_1 \cup (L_1 \cap \mathbb{R}^2_{\geq 0} + \{(\lambda, \lambda)^\intercal \mid \lambda \in \mathbb{R}_{\geq 0}\})$ and obtain a v_0 as desired, see Figure 3b. The latter inclusion is since supporting lines of the convex set K' through a boundary point $(1,0)^\intercal, (0,1)^\intercal \in L_1$ are constrained by $(0, \pm 1)^\intercal, (\pm 1, 0)^\intercal \in K'$. ◀

We can constructively apply this characterisation e.g. to norms that have polygons as sublevel sets. Such norms generalise the 1-norm, which allows exact CDTW algorithms [3], and can be used to approximate CDTW under the unwieldy 2-norm, as proposed in [5].

▶ **Corollary 3.3.** *If $K := S_{\leq 1}(\|\cdot\|)$ is a polygon with vertex sequence $\langle v_1, \ldots, v_k \rangle$, the metric d induced by $\|\cdot\|$ on $\mathbb{R}^2$ guarantees positively sloped valleys computable in time $O(\log(k))$, or in time $O(1)$ given $K = \psi(R)$ for a regular polygon $R \subseteq \mathbb{R}^2$ and a linear map $\psi\colon \mathbb{R}^2 \to \mathbb{R}^2$.*

Evaluating a polygonal norm via the approach in [8] and finding valleys via Theorem 3.2 are both possible through binary search on the vertices of the (necessarily convex) polygon [7], as indicated in Figure 3c. If the polygon is regular, we can compute the result directly.

Acknowledgements. The authors thank André Schulz for a helpful discussion.

References

1 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(1–2):75–91, 1995. doi:10.1142/S0218195995000064.

2 Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, Cambridge, 7th edition, 2009. URL: https://web.stanford.edu/~boyd/cvxbook/.

3 Milutin Brankovic. *Graphs and Trajectories in Practical Geometric Problems.* PhD thesis, University of Sydney, 2022.

4 Milutin Brankovic, Kevin Buchin, Koen Klaren, André Nusser, Aleksandr Popov, and Sampson Wong. (k,l)-medians clustering of trajectories using Continuous Dynamic Time Warping. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, pages 99–110, New York, 2020. ACM. doi:10.1145/3397536.3422245.

5 Kevin Buchin, André Nusser, and Sampson Wong. Computing Continuous Dynamic Time Warping of time series in polynomial time. In *38th International Symposium on Computational Geometry*, pages 22:1–22:16, Dagstuhl, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. arXiv:2203.04531, doi:10.4230/LIPICS.SOCG.2022.22.

6 Maike Buchin. *On the Computability of the Fréchet Distance between Triangulated Surfaces.* PhD thesis, Freie Universität Berlin, 2007. URL: https://refubium.fu-berlin.de/handle/fub188/1909.

7 Bernard Chazelle and David P. Dobkin. Intersection of convex objects in two and three dimensions. *Journal of the ACM*, 34(1):1–27, 1987. doi:10.1145/7531.24036.

8 L. Paul Chew and Robert L. (Scot) Drysdale, III. Voronoi diagrams based on convex distance functions. In *Proceedings of the First Annual Symposium on Computational Geometry*, pages 235–244, New York, 1985. ACM. doi:10.1145/323233.323264.

9 Jean-Lou De Carufel, Amin Gheibi, Anil Maheshwari, Jörg-Rüdiger Sack, and Christian Scheffer. Similarity of polygonal curves in the presence of outliers. *Computational Geometry*, 47(5):625–641, 2014. doi:10.1016/J.COMGEO.2014.01.002.

10 Anil Maheshwari, Jörg-Rüdiger Sack, and Christian Scheffer. Approximating the integral Fréchet distance. *Computational Geometry*, 70–71:13–30, 2018. doi:10.1016/J.COMGEO.2018.01.001.

11 Günter Rote. Lexicographic Fréchet matchings. Extended abstract from the 30th European Workshop on Computational Geometry, 2014. URL: https://refubium.fu-berlin.de/handle/fub188/15658.

12 Helmut H. Schaefer and Manfred P. Wolff. *Topological Vector Spaces*. Springer, New York, 2nd edition, 1999. doi:10.1007/978-1-4612-1468-7.

Representing Graphs by Unions of Line Segments

Petr Chmel and Vít Jelínek

Computer Science Institute, Charles University
`{chmel,jelinek}@iuuk.mff.cuni.cz`

Abstract

We consider intersection representations of graphs, in which every vertex is represented by a union of k horizontal or vertical segments in the plane, with two vertices x and y being adjacent if and only if at least one segment representing x intersects a segment from the representation of y.

We may further restrict the representations considered, by forbidding intersections between parallel segments, or by forbidding intersections between the i-th segment in the representation of x and the j-th segment in the representation of y unless $i = j$. This results in four types of graph representations.

Each of these four types gives rise to a graph parameter defined as the smallest k for which a graph G admits a representation of the given type. We describe how these parameters relate to each other, and study their relationship to other known graph parameters.

1 Introduction

Interval graphs are one of the most fundamental graph classes. There are several natural ways to generalize interval graphs to larger classes. One possibility are the k*-interval* graphs [8, 6, 16], which are the intersection graphs in which each vertex is represented by a union of up to k intervals, all belonging to the same line. A closely related class are the k*-track* graphs [8, 9], whose representation is obtained by first choosing k parallel lines, and then associating to each vertex a union of k intervals, with each of the k chosen lines containing one of the intervals. Clearly, a graph has a k-track representation if and only if it is a union of k interval graphs on a common vertex set.

These representations give rise to two graph parameters, known as the *interval number* $\mathrm{interval}(G)$ and the *track number* $\mathrm{track}(G)$, defined as the smallest k such that G has a k-interval representation or a k-track representation, respectively. These two parameters differ for some bipartite graphs [17] and some planar graphs [14, 7, 4].

A different way to generalize interval graphs is to consider segments in two directions, say horizontal and vertical. This yields the class of 2-DIR graphs [12], which are the intersection graphs of horizontal and vertical segments. We say that a 2-DIR representation is *pure* if no two parallel segments in it intersect, so each intersection involves one horizontal and one vertical segment. The class of graphs that admit such a representation is denoted by PURE-2-DIR [12] (also known as grid intersection graphs [10]). Necessarily, all such graphs are bipartite.

In this paper, we consider graphs that generalize 2-DIR and PURE-2-DIR in a similar way as k-interval and k-track graphs generalize interval graphs. Specifically, we obtain the following four types of intersection representations, which also naturally induce their respective graph parameters.

▶ **Definition 1.1** (Impure-line representation)**.** An *impure-k-line representation* of a graph $G = (V, E)$ is a family of k-tuples of horizontal or vertical line segments $R = \{(R_1(v), \dots, R_k(v)) : v \in V\}$ such that $\{u, v\} \in E \Leftrightarrow \exists \ell, m \in \{1, \dots, k\} : R_\ell(u) \cap R_m(v) \neq \emptyset$.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.

▶ **Definition 1.2** (Impure-tile representation). An *impure-k-tile representation* of a graph $G = (V, E)$ is an impure-k-line representation $R = \{(R_1(v), \ldots, R_k(v)) \colon v \in V\}$ such that $\forall u, v \in V \colon \forall \ell, m \in \{1, \ldots, k\} \colon \ell \neq m \Rightarrow R_\ell(u) \cap R_m(v) = \emptyset$.

▶ **Definition 1.3** (Pure-line representation). A *pure-k-line representation* of a graph $G = (V, E)$ is an impure-k-line representation $R = \{(R_1(v), \ldots, R_k(v)) \colon v \in V\}$ such that $\forall u, v \in V, \forall \ell, m \in \{1, \ldots, k\}$, if $R_\ell(u) \cap R_m(v) \neq \emptyset$, then precisely one of $R_\ell(u), R_m(v)$ is a horizontal line segment.

▶ **Definition 1.4** (Pure-tile representation). A *pure-k-tile representation* of a graph $G = (V, E)$ is a pure-k-line representation $R = \{(R_1(v), \ldots, R_k(v)) \colon v \in V\}$ such that $\forall u, v \in V \colon \forall \ell, m \in \{1, \ldots, k\} \colon \ell \neq m \Rightarrow R_\ell(u) \cap R_m(v) = \emptyset$.

We remark that for a fixed k and a graph G on n vertices, the above representations require only $\mathcal{O}(\log n)$ bits per vertex, thus yielding an efficient implicit representation [11].

▶ **Definition 1.5** (Parameters pure-tile, pure-line, impure-tile, impure-line). Given a graph G, we define the following four parameters:

- $\mathrm{il}(G)$ is the least k such that G has an impure-k-line representation,
- $\mathrm{it}(G)$ is the least k such that G has an impure-k-tile representation,
- $\mathrm{pl}(G)$ is the least k such that G has a pure-k-line representation,
- $\mathrm{pt}(G)$ is the least k such that G has a pure-k-tile representation.

We will first focus on the relationships among these four parameters. Among other results, we completely characterize the cases when one of the four parameters can be bounded as a function of another. This is the topic of Section 2. Next, in Section 3, we present results showing how these four parameters relate to other natural quantities, such as the degeneracy, the maximum degree, the thickness, the arboricity or the number of vertices of the represented graph.

Due to space constraints, we omit most of the proofs.

2 Bounds on our parameters

From the definitions, we directly obtain the following relationships between the four parameters.

▶ **Observation 2.1.** *For any graph G,* $\mathrm{pt}(G) \geq \mathrm{pl}(G) \geq \mathrm{il}(G)$ *and* $\mathrm{pt}(G) \geq \mathrm{it}(G) \geq \mathrm{il}(G)$.

▶ **Lemma 2.2.** *For any graph G,* $\mathrm{il}(G) \leq \mathrm{pl}(G) \leq 2 \cdot \mathrm{il}(G)$.

We also note that the constant 2 in the lemma is the best possible, as $\mathrm{il}(K_3) = 1$ because it is an interval graph, but $\mathrm{pl}(K_3) > 1$ as by the characterization of pure-1-line graphs due to Asinowski et al. [1], all graphs G with $\mathrm{pl}(G) = 1$ must be bipartite which K_3 is not.

By the previous results, we see that a graph class can fall into just four possible cases based on the boundedness of the parameters. These are as follows.

(I) The four parameters $\mathrm{il}, \mathrm{it}, \mathrm{pl}, \mathrm{pt}$ are all bounded.
(II) The parameters $\mathrm{il}, \mathrm{it}, \mathrm{pl}$ are bounded, while pt is unbounded.
(III) The parameters il, pl are bounded, while pt and it are unbounded.
(IV) The four parameters $\mathrm{il}, \mathrm{it}, \mathrm{pl}, \mathrm{pt}$ are all unbounded.

We call these types Type-I to Type-IV respectively. It is easy to observe that Type-I and Type-IV classes exist. For a Type-I class, we may consider the class of graphs with maximum degree 1, for which all the parameters are clearly equal to one. To obtain a Type-IV class, we may consider any class with $2^{\Theta(n^2)}$ graphs on n vertices, as having any parameter bounded by a constant would imply that there are at most $2^{\mathcal{O}(n \log n)}$ graphs on n vertices in the class, as the representations require only $\mathcal{O}(n \log n)$ bits.

Next, we exhibit Type-II and Type-III graph classes. For Type-II, we consider the class of complete graphs.

▶ **Lemma 2.3.** *Given a complete graph K_n on $n \geq 2$ vertices,* $\mathrm{pt}(K_n) = \lceil \log_2(n) \rceil$.

▶ **Observation 2.4.** *Given a complete graph K_n on $n \geq 2$ vertices,* $\mathrm{pl}(K_n) \leq 2$.

The two results together with the facts that $\mathrm{it}(K_n) = 1, \mathrm{il}(K_n) = 1$ as all complete graphs are interval graphs show that complete graphs indeed form a Type-II class. Moreover, we can extend this to show that any Type-II class must have an unbounded clique number $\omega(G)$.

▶ **Lemma 2.5.** *For any graph G,* $\mathrm{it}(G) \leq \mathrm{pt}(G) \leq \omega(G) \cdot \mathrm{it}(G)$.

In fact, this also implies a characterization of Type-II classes.

▶ **Corollary 2.6.** *If a class of graphs $\mathcal{G}$ has a bounded impure-tile parameter, then it is Type-II if and only if its clique number is unbounded, and it is Type-I otherwise.*

Having constructed a class of Type-II, we continue with exhibiting a class of Type-III. In this case, we use the class of complete multipartite graphs.

▶ **Lemma 2.7.** *For any $K \geq 3$ there exists a complete multipartite graph G_K such that* $\mathrm{it}(G_K) > K$ *and* $\mathrm{pl}(G_K) \leq 2$. *In particular, the impure-tile parameter may be unbounded while pure-line is bounded.*

▶ **Lemma 2.8.** *For any graph G with $\chi(G) \geq 2$, we have* $\mathrm{it}(G) \leq \lceil \log_2(\chi(G)) \rceil \cdot (\mathrm{il}(G))^2$.

Proof. We start by decomposing the graph G into $\lceil \log_2(\chi(G)) \rceil$ bipartite graphs. Let φ be a $\chi(G)$-coloring of G. We split the complete graph $K_{\chi(G)}$ into $\lceil \log_2(\chi(G)) \rceil$ bipartite graphs $H_1, \dots, H_{\lceil \log_2(\chi(G)) \rceil}$ on vertices $[\chi(G)]$. We then blow up each of the graphs H_i into a graph G_i with the vertex set $V(G)$ with two vertices $u, v \in V(G)$ forming an edge in the graph G_i if and only if $\{\varphi(u), \varphi(v)\} \in E(H_i)$ and $\{u, v\} \in E(G)$.

Next, we build an impure-k^2-tile representation from an impure-k-line representation R of a bipartite graph G_i partitioned into $A \dot\cup B$. For every pair $(\alpha, \beta) \in [k] \times [k]$, we create a tile that contains the representations $R_\alpha(u)$ for all $u \in A$ and $R_\beta(v)$ for all $v \in B$.

Given an edge $\{u, v\}$ with $u \in A, v \in B$, by definition there exists a pair of line segments $R_a(u), R_b(v)$ such that $R_a(u) \cap R_b(v) \neq \emptyset$. The edge is then realized by the intersection in the tile corresponding to the pair (a, b). Moreover, every pair of vertices that is not connected by an edge has no intersections between any two representing line segments in the impure-k-line representation, and therefore we cannot create the edge in the impure-k^2-tile representation.

Applying the construction on the $\lceil \log_2(\chi(G)) \rceil$ bipartite graphs G_i yields the theorem. ◀

Similarly to Type-II classes, the lemma implies a characterization of Type-III classes.

▶ **Corollary 2.9.** *If a class of graphs $\mathcal{G}$ has bounded impure-line parameter and bounded clique number, then it is Type-III if and only if its chromatic number is unbounded, and it is Type-I otherwise.*

3 Our parameters versus known parameters

Having defined new graph parameters, it is natural to relate these to the already known ones, and we do precisely that.

3.1 Degeneracy and maximum degree

We start with degeneracy and maximum degree.

▶ **Lemma 3.1.** *If G is a d-degenerate graph, then* $\mathrm{pt}(G) \leq d$*.*

We can use the 2-factor theorem to obtain similar results to Lemma 3.1 for graphs with bounded maximum degree in Theorem 3.3. If we restrict our attention to bipartite graphs, we can improve the bound for $\mathrm{pt}(\cdot)$ slightly in Theorem 3.2.

▶ **Theorem 3.2.** *If G is a bipartite graph with maximum degree Δ, then* $\mathrm{pt}(G) \leq \lceil \Delta/2 \rceil$*.*

▶ **Theorem 3.3.** *If G is a graph with maximum degree Δ, then* $\mathrm{it}(G) \leq \lceil \Delta/2 \rceil$ *and* $\mathrm{pt}(G) \leq \lceil (\Delta+1)/2 \rceil$*.*

Note that already for $\Delta = 3$ the bounds in Theorems 3.2 and 3.3 are tight since, by an argument dating back to Sinden [15], the subdivision of $K_{3,3}$ is not a string graph and hence also not a 2-DIR graph.

However, for larger Δ, we can no longer provide tight examples. This is because our explicit constructions of graphs with large $\mathrm{it}(\cdot)$ use Ramsey-type arguments, and the resulting graphs tend to have high degrees.

We saw that bounded maximum degree implies bounded $\mathrm{pt}(\cdot)$, and therefore for any Δ, the class of all graphs with maximum degree at most Δ is of Type I. On the other hand, our next two results show that the weaker assumption of bounded chromatic number does not imply the boundedness of any of our four parameters, since even the class of bipartite graphs is of Type IV. The proof is constructive, using a Ramsey-type argument. Later, in Theorem 3.10, we also give a nonconstructive counting argument yielding the same result.

▶ **Theorem 3.4.** *For all $d \geq 2$, there exists a bipartite graph G_d with* $\mathrm{it}(G) \geq d$*.*

▶ **Corollary 3.5.** *For all $d \geq 2$, there exists a bipartite graph G_d with* $\mathrm{il}(G) \geq d$*.*

We remark that Mustaţă and Pergel [13] showed that it is NP-hard to distinguish a PURE-2-DIR graph from a graph which is not a string graph, even when the input graph has degree bounded by 8. Therefore, any class of graphs “sandwiched” between PURE-2-DIR and string graphs is NP-hard to recognize even on graphs of degree at most 8. In contrast, Theorem 3.3 shows that such a hardness result cannot be extended to general impure-k-tile graphs for $k \geq 4$, or pure-k-tile graphs for $k \geq 5$, since these classes already contain all the graphs of degree at most 8.

3.2 Graph thickness, arboricity, interval and track numbers

We now relate our parameters to graph parameters which describe the decomposition of the original graph into graphs from a restricted class. We start with graph thickness, the minimum number of planar graphs the original graph can be decomposed to. By Lemma 3.1 and the fact that planar graphs are 5-degenerate, we know that for any planar G, $\mathrm{pt}(G) \leq 5$. However, we can do even better by using better representations of bipartite planar graphs [3].

▶ **Theorem 3.6** (de Fraysseix, Ossona de Mendez, and Pach [3])**.** *Any bipartite planar graph is a contact intersection graph of horizontal and vertical line segments.*

▶ **Theorem 3.7.** *For any planar graph G,* $\mathrm{pt}(G) \leq 2$.

Proof. By the four color theorem, let us have a 4-coloring $\varphi : V(G) \to \{1,2,3,4\}$. We decompose the edges G into two bipartite graphs G_1, G_2 based on the coloring as follows: $G_1 = (V(G), \{e \in E(G) : \varphi[e] = \{1,3\} \vee \varphi[e] = \{2,4\}\})$, $G_2 = (V(G), E(G) \setminus E(G_1))$.

By Theorem 3.6, both G_1 and G_2 are contact intersection graphs of horizontal and vertical line segments. Therefore, G has a pure-2-tile representation. ◀

We also immediately notice that this is tight as there are graphs that require two tiles: in particular, the complete graph on four vertices K_4, which by Lemma 2.3 requires at least two tiles. Moreover, this implies that for any G, $\mathrm{pt}(G)$ cannot be much larger than $\mathrm{thickness}(G)$.

▶ **Corollary 3.8.** *For any graph G,* $\mathrm{pt}(G) \leq 2 \cdot \mathrm{thickness}(G)$.

However, it is known that the thickness of a complete bipartite graph $K_{n,n}$ is $\lceil \frac{n^2}{4n-4} \rceil$ [2], and yet $\mathrm{pt}(K_{n,n}) = 1$, so the two parameters can be far from each other in the other direction.

Similarly, we consider arboricity. There, it suffices to observe that every forest is 1-degenerate, and thus, by Lemma 3.1, we have the following corollary.

▶ **Corollary 3.9.** *For any graph G,* $\mathrm{pt}(G) \leq \mathrm{arboricity}(G)$.

Again, the two parameters can be quite far from each other, and this can, again, be proved using the complete bipartite graph $K_{n,n}$: every forest can cover at most $2n - 1$ edges, and there are in total n^2 edges in the graph, thus $\mathrm{arboricity}(K_{n,n}) \geq \Omega(n)$, while $\mathrm{pt}(K_{n,n}) = 1$.

Another decomposition parameter is the edge-chromatic number, which is equal to the smallest number of matchings into which a graph can be decomposed. Since this parameter is equal to $\Delta(G)$ or $\Delta(G) + 1$, Theorem 3.3 shows that graphs of bounded edge-chromatic number have bounded $\mathrm{pt}(\cdot)$, while the converse does not hold, as shown by the example of $K_{n,n}$.

Finally, we consider the interval and track numbers. These two parameters are incomparable to pure-tile, as $\mathrm{interval}(K_n) = \mathrm{track}(K_n) = 1$ yet $\mathrm{pt}(K_n) = \lceil \log_2(n) \rceil$, while on the other hand, $\mathrm{pt}(K_{n,n}) = 1$ while $\mathrm{track}(K_{n,n}) \geq \mathrm{interval}(K_{n,n}) = \lceil \frac{n+1}{2} \rceil$ by a result of Griggs [5].

It is also clear that $\mathrm{it}(G) \leq \mathrm{track}(G)$ and $\mathrm{il}(G) \leq \mathrm{interval}(G)$, which by Lemma 2.2 implies $\mathrm{pl}(G) \leq 2 \cdot \mathrm{interval}(G) \leq 2 \cdot \mathrm{track}(G)$.

3.3 Number of vertices

Finally, we show bounds on the parameters with respect to the number of vertices of the graphs. We start with a lower bound, showing that there exist bipartite n-vertex graphs with impure-line representations requiring $\Omega(n/\log n)$ lines using a counting argument.

▶ **Theorem 3.10.** *There exists $n_0 \in \mathbb{N}$ such that $\forall n \geq n_0$, there exists a bipartite graph G with n vertices such that* $\mathrm{il}(G) \in \Omega(\frac{n}{\log n})$.

This bound also holds for the three remaining parameters, as we showed the impure-line parameter to be less than or equal to the remaining parameters.

Next, we complement this result with an upper bound, though a gap remains.

▶ **Lemma 3.11.** *Every graph on n vertices has a pure-$\lceil \frac{n}{2} \rceil$-tile representation.*

Again, this also holds for the three remaining types of representations, as a pure-tile representation is the most restricted type of the four representations we consider.

References

1 Andrei Asinowski, Elad Cohen, Martin C. Golumbic, Vincent Limouzy, Marina Lipshteyn, and Michal Stern. Vertex intersection graphs of paths on a grid. *J. Graph Algorithms Appl.*, 16(2):129–150, 2012. doi:10.7155/jgaa.00253.

2 Lowell W. Beineke, Frank Harary, and John W. Moon. On the thickness of the complete bipartite graph. *Mathematical Proceedings of the Cambridge Philosophical Society*, 60(1):01–05, 1964. doi:10.1017/S0305004100037385.

3 Hubert de Fraysseix, Patrice Ossona de Mendez, and János Pach. Representation of planar graphs by segments. Technical report, North-Holland, 1994.

4 Daniel Gonçalves and Pascal Ochem. On star and caterpillar arboricity. *Discrete Mathematics*, 309(11):3694–3702, 2009. 7th International Colloquium on Graph Theory. doi:10.1016/j.disc.2008.01.041.

5 Jerrold R. Griggs. Extremal values of the interval number of a graph, II. *Discrete Mathematics*, 28(1):37–47, 1979. doi:10.1016/0012-365X(79)90183-3.

6 Jerrold R. Griggs and Douglas B. West. Extremal values of the interval number of a graph. *SIAM Journal on Algebraic Discrete Methods*, 1(1):1–7, 1980. doi:10.1137/0601001.

7 Guillaume Guégan, Kolja Knauer, Jonathan Rollin, and Torsten Ueckerdt. The interval number of a planar graph is at most three. *Journal of Combinatorial Theory, Series B*, 146:61–67, 2021. doi:10.1016/j.jctb.2020.07.006.

8 András Gyárfás and Jenö Lehel. A Helly-type problem in trees. *Combinatorial Theory and its applications*, 4:571–584, 1970.

9 András Gyárfás and Douglas West. Multitrack interval graphs. *Congr. Numerantium*, 109:109–116, 1995.

10 I. B-Arroyo Hartman, Ilan Newman, and Ran Ziv. On grid intersection graphs. *Discrete Mathematics*, 87(1):41–52, 1991. doi:10.1016/0012-365X(91)90069-E.

11 Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. *SIAM Journal on Discrete Mathematics*, 5(4):596–603, 1992. doi:10.1137/0405049.

12 Jan Kratochvíl and Jiří Matoušek. Intersection graphs of segments. *Journal of Combinatorial Theory, Series B*, 62(2):289–315, 1994. doi:10.1006/jctb.1994.1071.

13 Irina Mustaţă and Martin Pergel. What makes the recognition problem hard for classes related to segment and string graphs? *CoRR*, abs/2201.08498, 2022. arXiv:2201.08498.

14 Edward R. Scheinerman and Douglas B. West. The interval number of a planar graph: Three intervals suffice. *Journal of Combinatorial Theory, Series B*, 35(3):224–239, 1983. doi:10.1016/0095-8956(83)90050-3.

15 F. W. Sinden. Topology of thin film RC circuits. *The Bell System Technical Journal*, 45(9):1639–1662, 1966. doi:10.1002/j.1538-7305.1966.tb01713.x.

16 William T. Trotter Jr. and Frank Harary. On double and multiple interval graphs. *Journal of Graph Theory*, 3(3):205–211, 1979. doi:10.1002/jgt.3190030302.

17 Douglas B. West and David B. Shmoys. Recognizing graphs with fixed interval number is NP-complete. *Discrete Applied Mathematics*, 8(3):295–305, 1984. doi:10.1016/0166-218X(84)90127-6.

Guarding an orthogonal polygon with the minimum number of sliding cameras placed on the vertical edges

Nazanin Hadiniya[1], **Mohammad Ghodsi**[2], **and Wolfgang Mulzer**[3]

1 **Department of Computer Engineering, Sharif University of Technology of Tehran, Iran**
nazanin.hadiniya73@sharif.edu

2 **School of Computer Science, Institute for Research in Fundamental Sciences (IPM) of Tehran, Iran**
ghodsi@sharif.edu

3 **Department of Mathematics and Computer Science, Free University of Berlin, Germany**
mulzer@inf.fu-berlin.de

Abstract

In this paper, we study the problem of guarding an n-vertex orthogonal polygon P with unlimited-vision sliding cameras that can move back and forth along an orthogonal line segment s. These cameras can see a point p in the polygon P if and only if there exists a line segment completely in P and containing p that is perpendicular to s. In this paper, we restrict camera placement to the vertical edges of the polygon. So, each camera can move along a vertical edge of the polygon and covers the interior part of the polygon, which is perpendicular to it. To solve the problem, we model it as a graph and demonstrate that solving the vertex cover problem on this graph is equivalent to solving our original problem. Although the vertex cover problem is NP-hard in general, we prove that our graph is bipartite, allowing it to be solved in polynomial time.

1 Introduction

The art gallery problem is a well-known problem in computational geometry, and its goal is to completely guard an interior part of a polygon [14]. Some variants of this problem are NP-hard. For example, vertex guarding (where all guards lie on the vertices) and point guarding (where each guard can be anywhere in the polygon) are NP-hard even for orthogonal polygons [1, 13, 14]. Some limits on the number of guards have been found. For example, $\lfloor \frac{n}{3} \rfloor$ guards are sufficient and sometimes necessary to cover an arbitrary polygon. For orthogonal polygons, at most $\lfloor \frac{n}{4} \rfloor$ guards are needed [14]. There are also many different approximation algorithms for these problems, such as [7].
The problem of guarding orthogonal polygons with sliding cameras is a newer version of the art gallery problem, initially introduced by Katz and Morgenstern in 2011 [11]. In this version, each camera moves back and forth along an orthogonal line segment s and guards the interior regions of the polygon that are perpendicular to s. The objective of the problem is minimizing either the number of cameras or the total length of the segments along which the cameras move. Later, a polynomial-time algorithm was proposed to minimize the total length; however, so far, no polynomial-time algorithm has been developed for minimizing the number of cameras.

Contributions. In this paper, we give an exact algorithm to solve the problem of guarding an orthogonal polygon P with sliding cameras placed on the vertical edges of P. Our goal

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.

is to minimize the number of cameras needed and, for this, we reduce it to a vertex cover problem like [8]. Figure 1 shows the minimum number of cameras in four different versions of the problem. In Figure 1(a) and Figure 1(b), both vertical and horizontal sliding cameras are allowed, but in Figure 1(a) they can only be on the edges of the polygon, and in Figure 1(b) they can be anywhere in the polygon. In Figure 1(c) and Figure 1(d), sliding cameras are placed exclusively on the vertical or horizontal edges of the polygon.

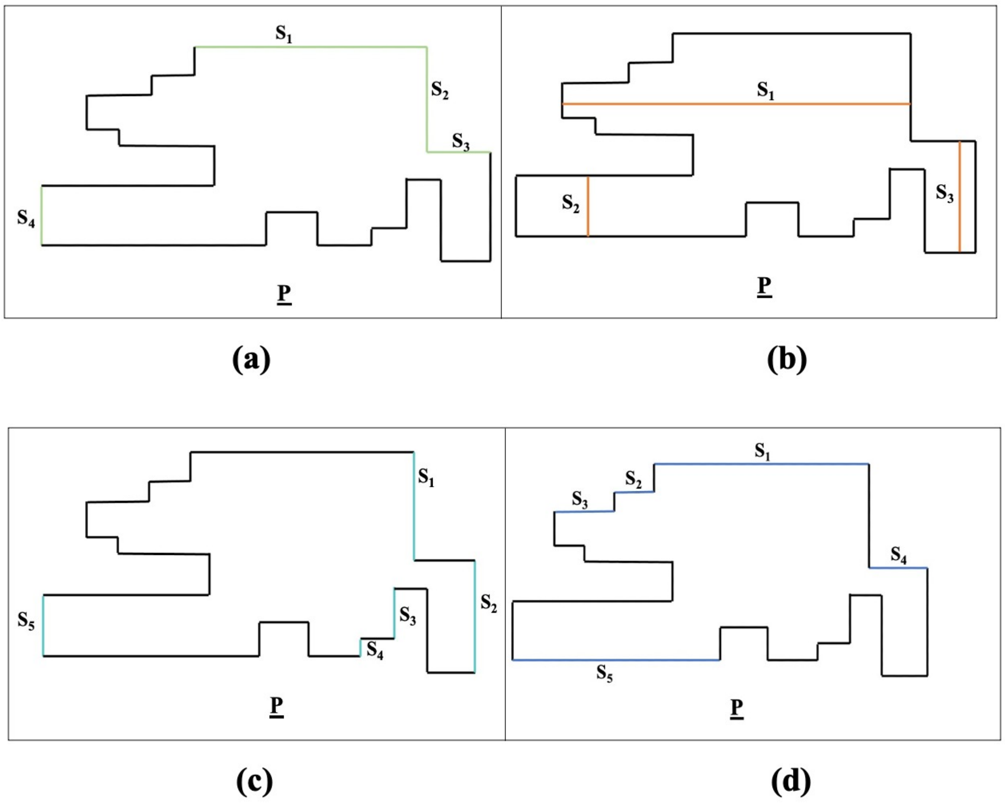


Figure 1 An orthogonal polygon and the minimum number of sliding cameras to cover it in different versions: (a) Sliding cameras can be placed vertically and horizontally on the edges of the polygon. (b) Sliding cameras can be placed vertically and horizontally anywhere in the polygon. (c) Sliding cameras can be placed on vertical edges of the polygon. (d) Sliding cameras can be placed on horizontal edges of the polygon.

In the rest of the paper, we first review the relevant background in Section 2 and then in Section 3, we describe the partitioning of the polygon and its representation as a graph. Finally, in Section 4, we show that solving vertex cover problem in this graph is equivalent to the solution of our problem.

2 Related works

Guarding with sliding cameras was first introduced by Katz and Morgenstern [11]. They have worked on a special case of the problem and gave a polynomial time algorithm for solving it. To do this, they transformed the problem into an equivalent problem by modeling it to a chordal graph and finding the minimum number of cliques on this graph. For the general case of the problem where vertical and horizontal guards are allowed, they gave a 3-approximation algorithm for x-monotone orthogonal polygons (A polygon P is x-monotone if every vertical line intersects the polygon boundaries at most twice). At the end, they presented the problem with the objective of minimizing the total length of the sliding cameras. This problem was solved by Durocher and Mehrabi [8]. They have shown that it is solvable in polynomial time by reducing it to the minimum weight vertex cover problem in bipartite graphs. This problem is polynomial even for orthogonal polygons with holes, but for finding the minimum number of sliding cameras in orthogonal polygons with holes, they have proved it is NP-hard. In 2014, Durocher and Mehrabi [9] gave an approximation algorithm for finding the minimum number of sliding cameras with 3.5 approximation factor and $\mathcal{O}(n^{\frac{5}{2}})$ time order. Later, in 2017, they found out that this factor was incorrect. Biedl et al. gave an $\mathcal{O}(1)$-approximation algorithm for the problem. Besides, for orthogonal polygons with holes, they proved that it is NP-hard even for the case that only horizontal (vertical) cameras are allowed. Their algorithm works for polygons with holes, too. In 2017, DeBerg et al. [7] presented a polynomial algorithm for a special case where polygons are x-monotone. Moreover, their algorithm achieves $\mathcal{O}(n)$ time complexity for orthogonal polygons whose dual graph, derived from vertical decomposition, forms a path. Later, Bandyapadhyay and Basu Roy [3] have investigated approximation algorithms for the problems of finding minimum number of horizontal sliding cameras and minimum number of horizontal and vertical sliding cameras. For these problems, they proved the existence of a PTAS algorithm using the local-search framework. Also, in the case of horizontal sliding cameras, their proof is also valid for polygons with holes. In 2021, Biedl and Mehrabi [4] investigated different models of guarding orthogonal polygons. They showed that for each of the rectangular, staircase, periscope, and sliding cameras guarding, if the orthogonal polygon is with bounded tree-width, the time order of their algorithm is polynomial. Another form of the problem is k-transmitters sliding cameras problem. For this problem, Biedl et al. [5] showed that finding the minimum number of sliding cameras in any orthogonal polygon and for every $k > 0$ is NP-hard and gave an $\mathcal{O}(1)$-approximation algorithm for finding the minimum number of horizontal sliding cameras and the case in which both horizontal and vertical sliding cameras are allowed.

3 Modeling the polygon as a graph

In our problem, we get an orthogonal polygon P as our input and want to place m sliding cameras $S = \{s_1, s_2, \ldots\ s_m\}$ on some of vertical edges of this polygon which can cover the entire interior part of P by moving on their edge e_i. Depending on the edge, each camera moves back and forth along it, covering the area to its right or left, which is inside the polygon and perpendicular to it. The goal is to minimize m, which means minimizing the number of cameras needed.

3.1 Partitioning the Polygon P

To solve our problem, we divide our polygon P into rectangular partitions through its reflex vertices. To do this, use the same way as Katz and Morgenstern did in [11]. Extend the two edges of P incident to every reflex vertex inward until they hit the boundaries of P. Figure 2 shows a polygon P and its reflex vertices. After extending its reflex vertices, the polygon divided into several rectangles. The Set $Rec(P)$ represents these rectangles.

▶ **Lemma 3.1.** *The number of rectangles created after partitioning the polygon P with n vertices and r reflex vertices is $\mathcal{O}(n^2)$.*

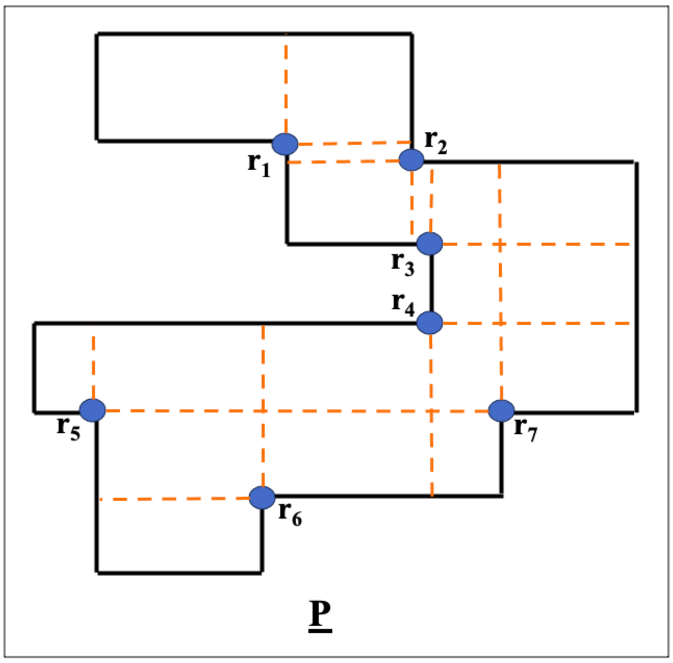


Figure 2 Partitioning of polygon P through its reflex vertices

Proof. As [12] showed, there are $\frac{n-4}{2}$ reflex vertices in each orthogonal polygon P. So, we have $\mathcal{O}(n)$ reflex vertices. Furthermore, it was proven in [2] that the number of rectangles created after partitioning through reflex vertices is at most r^2+r+1, which means $\mathcal{O}(n^2)$. ◀

3.2 Construction of the graph G_P

To construct the equivalent undirected graph G_P for our problem, we utilize the sets $E(P)$ and $Rec(P)$. The set $E(P)$ represents the vertical edges of the polygon P, while the set $Rec(P)$ as previously defined, denotes the rectangles formed after partitioning the polygon. Each e_i in $E(P)$ corresponds to a vertex in the graph G_P. Thus, for each e_i in $E(P)$, we create a vertex v_i in the graph G_P. Therefore, graph G_P has $\mathcal{O}(n)$ vertices. Now, two vertices v_i and v_j are adjacent in G_P if and only if their corresponding edges e_i and e_j in P see a common rectangle R in $Rec(P)$. It is evident that each rectangle R in $Rec(P)$ is visible from exactly two edges, e_i and e_j of P. In fact, based on the cameras' field of view, each rectangle is observed by one camera from above and one from below. As we traverse each rectangle R in $Rec(P)$, if the edge between v_i and v_j of G_P was inserted before, no additional edge is added; otherwise, we insert an edge between these vertices.
Figure 3 shows a polygon P and the graph G_P constructed from P.

▶ **Observation 3.2.** *After constructing the graph G_P, each rectangle in the set $REC(P)$ corresponds to an edge in G_P. Since each rectangle can be covered by exactly two edges of P, and during the construction of graph G_P, an edge is inserted between two vertices if their corresponding edges in P see a common region; this means that every rectangle contributes an edge to G_P. On the other hand, there might be more than one common rectangle between two edges of P; in this case, an edge between two vertices of G_P can represent more than one rectangle of P.*

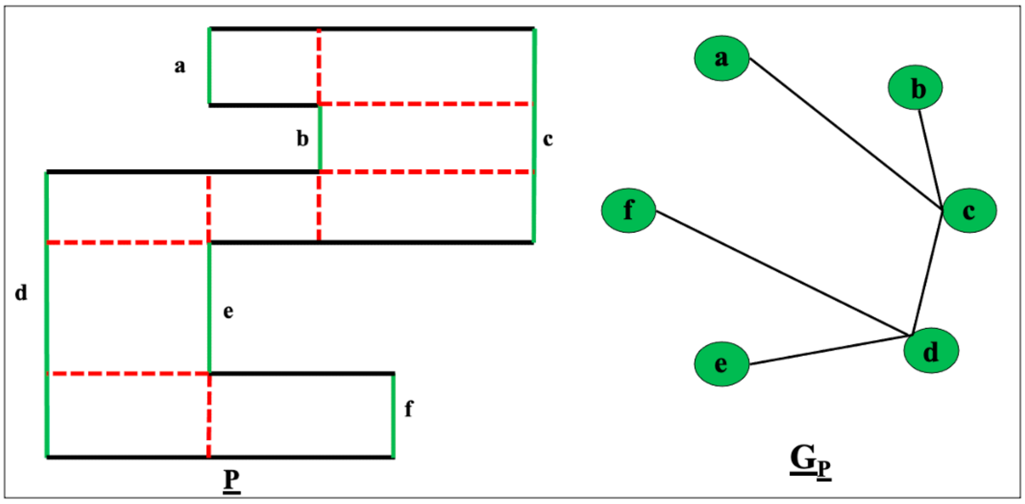


Figure 3 an orthogonal polygon P and its equivalence Graph G_P in the unlimited-vision sliding cameras problem

4 An algorithm for solving the problem of minimizing the number of vertical sliding cameras

The Vertex Cover problem is generally NP-hard; however, it has been proven [6] that the problem can be solved in polynomial time for bipartite graphs due to its equivalence to the Maximum Matching problem. Therefore, we want to show that G_P is a bipartite graph, and by solving vertex cover, our problem is solved.

▶ **Definition 4.1.** For a given graph $G = (V, E)$, the vertex cover problem is to find a subset of its vertices V such that every edge in E has at least one endpoint in this subset [10].

After constructing the graph G_P, we prove that solving vertex cover in graph G_P is equal to solving our problem and then show that graph G_P is a bipartite graph.

▶ **Theorem 4.2.** *The problem of guarding an orthogonal polygon P with the minimum number of vertical sliding cameras placed on the edges of P reduces to the vertex cover problem in graph G_P*

Proof. Let C_G be any vertex cover of G_P and let S_P be a guarding of P. Now, we should show C_G is a minimum vertex cover of G_P if and only if S_P is an optimal guarding of P.
For the first part, suppose any vertex cover C_G and we find a guarding S_P of P by it. For each edge (v_i, v_j) if v_i is a vertex in C_G we choose the edge e_i of P. Otherwise, we choose e_j of P. Since at least one vertex of each edge in G_P is in C_G, by Observation 3.2 we conclude that each rectangle of the polygon P is covered by at least one sliding camera. This shows that the number of vertices in every C_G is equal to the number of sliding cameras in S_P. So, the minimum vertex cover is equal to the minimum number of sliding cameras needed to cover P.
For the second part, suppose any guarding S_P in polygon P. Similar to the previous part we find a vertex cover in G_P. For each edge e_i in the S_P, choose the equivalent vertex v_i in G_P and put it in the C_G. We know that all the rectangles $Rec(P)$ are covered by S_P and by the Observation 3.2, we know that any edge in graph G_P is equal to one or more rectangles of polygon P. So, when all these rectangles are covered, all the edges of the graph G_P have

at least one incident vertex to it in the C_G and C_G is a vertex cover for G_P. As a result, guarding S_P is a vertex cover of G_P and the minimum number of cameras needed to cover the polygon P is equal to the minimum vertex cover of G_P. By these two parts, we conclude that our reduction is correct and the theorem is proved. ◀

▶ **Lemma 4.3.** *The graph G_P, constructed from the polygon P, is bipartite.*

Proof. We construct the graph by only vertical edges of the polygon P. There are two types of vertical edges in the polygon P: Edges whose inferior region is inside the polygon and edges whose superior region is inside the polygon. Two edges of the same type can't see a common area of the polygon. As a result, their equivalent vertices in the graph G_P aren't adjacent. So, vertices of each type belong to one side of the bipartite graph. ◀

5 Conclusion and future work

In this paper, we introduced a special case of guarding with sliding cameras where each camera moves back and forth through the edges of the polygon. We developed an exact algorithm for solving the problem of minimizing the number of vertical cameras needed to cover the polygon. To solve the problem, we used a reduction like [8] to the vertex cover problem in bipartite graphs. We can extend our solution, with some modifications, to the problem of guarding a polygon using the limited-vision version of sliding cameras. In this version, a parameter l is provided as input, representing the maximum distance each camera can cover. To solve the problem, we first determine whether the polygon can be fully covered by these cameras. Then, if possible, we draw parallel lines at a distance of l from the edges and extend them until they intersect the polygon's boundaries. Finally, we apply our algorithm to compute the minimum number of cameras required. However, in this version, certain rectangular regions may be visible from only a single edge of the polygon. As a result, those edges must be selected, which may lead to an increased in the number of sliding cameras compared to the previous version. We are also working on the problem of guarding orthogonal polygons using vertical and horizontal sliding cameras placed on the edges. For another future work, we can study the problem of guarding an orthogonal polygon using limited-vision sliding cameras that can be placed anywhere within the polygon. We conjecture that this problem is NP-hard.

References

1 Anders Aamand, Mikkel Abrahamsen, Peter MR Rasmussen, and Thomas D Ahle. Tiling with squares and packing dominos in polynomial time. *ACM Transactions on Algorithms*, 19(3):1–28, 2023.

2 António Leslie Bajuelos, Ana Paula Tomás, and Fábio Marques. Partitioning orthogonal polygons by extension of all edges incident to reflex vertices: lower and upper bounds on the number of pieces. In *Computational Science and Its Applications–ICCSA 2004: International Conference, Assisi, Italy, May 14-17, 2004, Proceedings, Part III 4*, pages 127–136. Springer, 2004.

3 Sayan Bandyapadhyay and Aniket Basu Roy. Effectiveness of local search for art gallery problems. In *Algorithms and Data Structures: 15th International Symposium, WADS 2017, St. John's, NL, Canada, July 31–August 2, 2017, Proceedings 15*, pages 49–60. Springer, 2017.

4 Therese Biedl and Saeed Mehrabi. On orthogonally guarding orthogonal polygons with bounded treewidth. *Algorithmica*, 83:641–666, 2021.

5 Therese Biedl, Saeed Mehrabi, and Ziting Yu. Sliding k-transmitters: hardness and approximation. *arXiv preprint arXiv:1607.07364*, 2016.

6 Norman Biggs, E Keith Lloyd, and Robin J Wilson. *Graph Theory, 1736-1936.* Oxford University Press, 1986.

7 Mark De Berg, Stephane Durocher, and Saeed Mehrabi. Guarding monotone art galleries with sliding cameras in linear time. *Journal of Discrete Algorithms*, 44:39–47, 2017.

8 Stephane Durocher and Saeed Mehrabi. Guarding orthogonal art galleries using sliding cameras: algorithmic and hardness results. In *Mathematical Foundations of Computer Science 2013: 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings 38*, pages 314–324. Springer, 2013.

9 Stephane Durocher and Saeed Mehrabi. A 3-approximation algorithm for guarding orthogonal art galleries with sliding cameras. In *International Workshop on Combinatorial Algorithms*, pages 140–152. Springer, 2014.

10 Richard M Karp. *Reducibility among combinatorial problems.* Springer, 2010.

11 Matthew J Katz and Gila Morgenstern. Guarding orthogonal art galleries with sliding cameras. *International Journal of Computational Geometry & Applications*, 21(02):241–250, 2011.

12 Joseph O'Rourke. An alternate proof of the rectilinear art gallery theorem. *Journal of Geometry*, 21:118–130, 1983.

13 Raphael M Robinson. Undecidability and nonperiodicity for tilings of the plane. *Inventiones mathematicae*, 12:177–209, 1971.

14 Jorge Urrutia. Art gallery and illumination problems. In *Handbook of computational geometry*, pages 973–1027. Elsevier, 2000.

A Robust Algorithm for Finding Triangles in Disk Graphs

Katharina Klost[1], Kristin Knorr[1], and Wolfgang Mulzer[1]

1 **Freie Universität Berlin**
{kathklost, knorrkri, mulzer}@inf.fu-berlin.de

Abstract

In a promise setting an algorithm utilizes the properties of a specified input domain to achieve a higher efficiency. The domain of the input instance needs to be correct to exclude errors. In contrast, a robust algorithm always terminates by either solving the given instance correctly or by reporting the instance not to be in the domain. In case of geometric intersection graphs, robust algorithms may use beneficial geometric properties. Hence, we are able to exclude input instances by identifying forbidden substructures. In this work, we present a robust algorithm for finding triangles in general disk graphs in linear time by considering planarity or crossing properties respectively.

1 Introduction

When considering algorithmic problems, we determine the running time in dependency of a specific input. We can often observe that some input classes are easier to solve than others if they have properties that simplify the given problem. We will refer to such input classes as *input domains*. An algorithm tailored to a specific input domain utilizes the advantages to gain efficiency. Algorithms in the so called *promise setting* are fault-prone if the input does not belong to the expected domain, e.g. the algorithm returns a wrong result due to false assumptions. However, it could be hard to test beforehand whether an input instance belongs to the expected input domain. *Robust algorithms* circumvent these difficulties:

▶ **Definition 1.1** (Raghavan and Spinrad [9]). A *robust* algorithm for a problem P on a domain C solves P correctly, if the input is from C. If the input is not in C, the algorithm may either produce a correct answer for P or report that the input is not in C.

In their work Raghavan and Spinrad [9] give a robust polynomial-time algorithm for the Clique-problem in unit disk graphs. Recently, Klost and Mulzer [7] reconsidered the concept of robust algorithms in the domain of intersection graphs. They present robust algorithms for finding triangles and the unweighted girth, that is the length of the shortest cycle, in unit disk graphs. They also give a robust algorithm for finding a triangle in a transmission graph.

Disk graphs are defined on a set $S \subseteq \mathbb{R}^2$ of n sites in the plane, where each site $s \in S$ has an associated radius r_s. The *disk graph $D(S)$ on S* is defined as $D(S) = (S, E)$, where $E = \{\{s,t\} \mid \|st\| \leq r_s + r_t\}$ with $\|st\|$ being the Euclidean distance between the sites s and t. This is equivalent to requesting that the disks defined by s and t intersect and thus the graph is undirected.

Given an abstract undirected graph $G = (V, E)$, we say that G is a disk graph, if there exists a set of sites S, such that $G = D(S)$. We say that the set S *realizes* the disk graph. Graph G is called a *unit disk graph* if there is a realization with sites that all have the same radius.

Klost and Mulzer also studied a directed variant of disk graph, called *transmission graphs*. Here two sites are connected by a directed edge st if t lies in the disk defined by s. In this paper we will however focus on disk graphs.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Determining if a given abstract graph is realizable as a disk graph is known to be $\exists\mathbb{R}$-complete [5] and therefore testing beforehand if a given graph is a disk graph is not feasible.

In this paper we will extend the recent results by a robust algorithm for finding triangles and computing the unweighted girth in the domain of general disk graphs.

2 Preliminaries

We assume that the input is an abstract unweighted undirected graph $G = (V, E)$, given in an adjacency list representation. Throughout the paper, we will denote by n the number of vertices and by m the number of edges of a given graph. Given a vertex v of a graph, we denote by $N(v)$ the set of vertices that are adjacent to v, and by $\deg(v) = |N(v)|$ the degree of v. In the adjacency list representation, a set of k neighbors of v can be reported in $O(k)$ time, and testing if two vertices u and v are adjacent takes $O(\min(\deg(v), \deg(u)))$ time. The *girth* of a graph G is the length of the shortest cycle in G.

In the proof below, it will be beneficial if all adjacency lists are sorted by a common order. With a given order of vertices, we are able to produce sorted adjacency lists in linear time with respect to vertices and edges.

▶ **Lemma 2.1.** *Let $G = (V, E)$ be an abstract graph in adjacency list representation and let π be a total order on V. Assume that π is represented in a way, such that the vertices can be iterated over in this order in linear time. Then we can generate a sorted copy of the adjacency lists according to this order in $O(n + m)$ time.*

Proof. We iterate over V in the order of π. When visiting the ith vertex v_i we pass through its adjacency list. For each vertex u in the adjacency list of v_i, we do the following: If we encounter u for the first time, we store a new adjacency list l'_u for u and add v_i as its only vertex. However, if the adjacency list l'_u already exists, i.e. we have seen u before, we insert v_i to the end of l'_u.

We can iterate over the vertices in $O(n)$ time. For each vertex we iterate over its adjacency list once with constant running time for each entry. Thus, the algorithm takes $O(n + m)$ overall time.

With this procedure, we add a vertex v_i to the end of the new adjacency lists l' exactly when it appears in π. Hence, a vertex v_i is inserted after all vertices with index $j < i$ and before all vertices with index $\ell > i$. The new adjacency lists now contain all adjacent vertices in the desired order. ◀

There are several algorithms known which check in linear time if an abstract graph is planar [1, 4]. These algorithms make use of Kuratowski's theorem which states that a finite graph is planar if and only if the graph does not contain a subgraph which is a subdivision of $\mathbf{K}_5$ or $\mathbf{K}_{3,3}$ [8]. Such a subdivision is known as a Kuratowski subgraph.

A given geometric realization of a disk graph G naturally induces an embedding of the graph into the plane. Let S be the sites realizing the graph. Then each vertex is embedded at the coordinate of the corresponding site and the edges are embedded as the straight lines between the sites. When it is clear from the context we will not differentiate between a vertex and its site in a realization.

Throughout the paper, we will need some basic properties on disk graphs which are originated from their geometry. The following result stated by KKMRSS [6] is a crucial

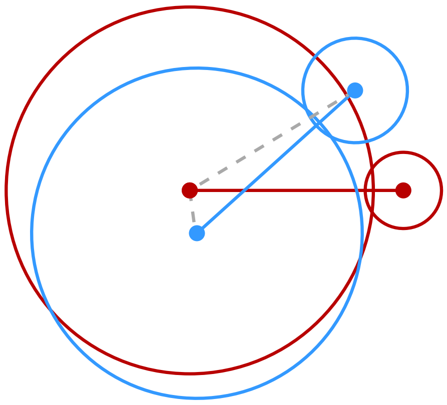


Figure 1 If two edges intersect, at least three of the disks corresponding to their endpoints have a common point.

ingredient for the correctness of our algorithms.[1]

▶ **Lemma 2.2** (KKMRSS [6]). *Let $D(S)$ be a disk graph on a set of sites S that is not plane, i.e., the embedding that represents each edge by a line segment between its endpoints has two segments that cross in their relative interiors. Then, the associated disks of three of the sites defining the edges intersect in a common point*

The proof builds on the observation that two intersecting disks form a lens that has to be intersected by at least one disk of a crossing edge, see Figure 1. For further details we refer to the arXiv version of the work by KKMRSS [6]. By contraposition, we get the following Corollary.

▶ **Corollary 2.3.** *If a (unit) disk graph does not contain a triangle, then it is planar*

Proof. The contraposition of Lemma 2.2 states that if the (unit) disk graph does not contain a triangle, the embedding of any realization is plane. Thus, the graph is planar. ◀

3 Finding Triangles in General Disk Graphs

In their previous work, Klost and Mulzer [7] described a robust algorithm to find triangles in unit disk graphs. Their algorithm makes use of the well known geometric property that unit disk graphs with at least one vertex with degree larger than five contain a triangle that uses this vertex. As often in intersection graph settings, the problem becomes more involved if we allow the size of the objects to vary. In the case of general disk graphs the radii of the disks may differ in size. Particularly, every star is a general disk graph (see Figure 2) and thus the property used by Klost and Mulzer is not applicable for general disk graphs. However, Corollary 2.3 holds for general disk graphs and can still be applied.

This is a similar situation to the geometric case considered by KKMRSS [6]. Similarly to their algorithm, we consider the case where G is planar separately from the non-planar case. As there are algorithms that find triangles and compute the girth in planar graphs in linear time [2], the first case can easily be handled. For the non-planar case, we need an

[1] Even though we cite the version of the lemma that is given by KKMRSS [6], let us mention that this fact has been known for a long time (e.g., [3]), also in more general contexts such as intersection graphs of pseudodisks (we would like to thank Shakhar Smorodinsky for pointing this out).

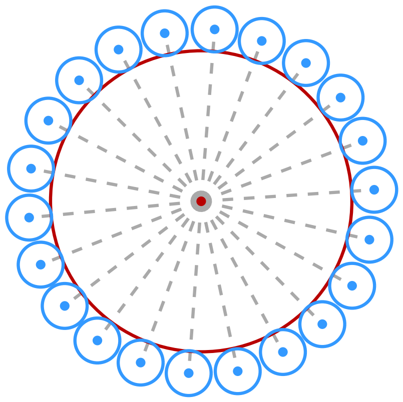


Figure 2 Every star can be realizied as a general disk graph.

approach tailored to the robust setting for general disk graphs. Corollary 2.3 implies either the existence of a triangle or the non-realizability as a disk graph. Our algorithm will make use of the existence of a subgraph that is a Kuratowski subdivision. Let G be a non-planar graph and $G_K = (V_K, E_K)$ be a subgraph of G that forms a Kuratowski subdivision. Let G_{iK} be the subgraph of G induced by V_K. The correctness of the algorithm will depend on the following lemma.

▶ **Lemma 3.1.** *Let G be a non-planar disk graph, let $G_K = (V_K, E_K) \subseteq G$ be a Kuratowski subdivision on G. Also let G_{iK} be the induced subgraph of V_K. Then there exists a triangle in G_{iK} that contains an edge of E_K.*

Proof. As G is a disk graph, there is a set S of sites realizing G. Consider the embedding induced by S. As by definition G_K is a non-planar subgraph of G, there is at least one pair of crossing edges $e_i, e_j \in E_K$ in its straight line embedding induced by S. See Figure 3 for an illustration. By Lemma 2.2, at least three of the vertices incident to e_i and e_j form a triangle. This directly implies that one of the edges e_i or e_j is an edge of the triangle and the statement follows. ◀

Lemma 3.1 is the main ingredient to solve the non-planar case in our robust algorithm. If the graph is not planar, we will explicitly check for each edge of E_K if it is part of a triangle. Details are given in the following theorem.

▶ **Theorem 3.2.** *There is a robust algorithm to find a triangle in a graph in the domain of general disk graphs that runs in $O(n + m)$ time.*

Proof. The Algorithm In the first step of the algorithm we run a planarity test, e.g. Boyer and Myrvold [1], on the graph G. If the test reports that the graph is planar, we run the algorithm by Chang and Lu [2] to find the girth on planar graphs. We return the triangle if the girth is three or the non-existence of a triangle otherwise.

If G is not planar, the planarity testing algorithm returns a Kuratowski subgraph $G_K = (V_K, E_K)$ of G which we can examine. Note that even if we knew a pair of crossing edges in some embedding of G_K this pair might not form a triangle in a realization of the disk graph. Thus, we need to explicitly check for all edges of G_K if they are part of a triangle. For this, note that the top-level array of the adjacency list representation of G induces an ordering. We use this ordering to compute an ordered adjacency list representation of G by applying Lemma 2.1. Let i be the index of v in the top level array. Then we store this index with the vertex object of v during the sorting.

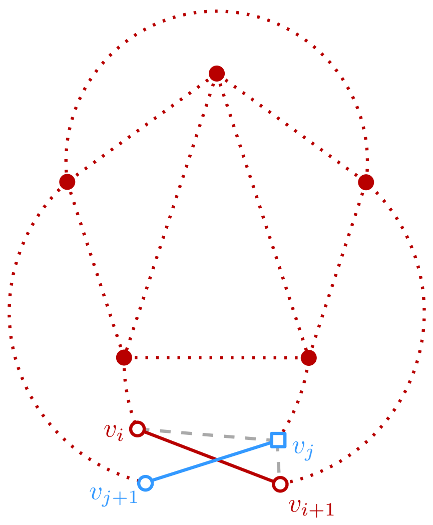


Figure 3 There is a pair of crossing edges in the embedding of G_K.

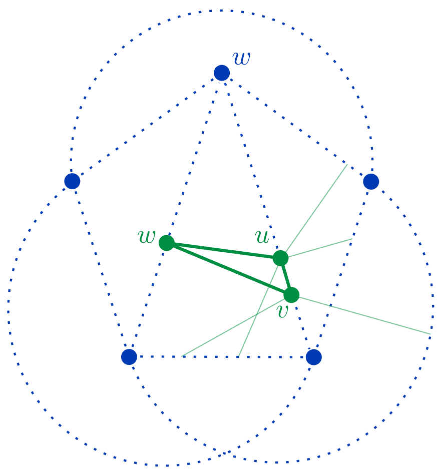


Figure 4 The vertices u, v form an edge in E_K and have a common neighbor. Thus, the triangle u, v, w is found. The green edges are edges in G_{iK}.

In order to test for triangles of the form described by Lemma 3.1, we iterate over all edges $\{u, v\} \in E_K$. Consider the adjacency lists of u and v. Traverse those lists in an interleaved fashion, always progressing in the list in which the index stored with the current vertex is smaller. If the same vertex w is found in both lists, stop and report the triangle u, v, w, see Figure 4. Otherwise, if no such triangle was found after testing all edges in E_K, report that G is not a disk graph.

Correctness The correctness of the planar case follows from the previous algorithms [6]. If a triangle is reported in the non-planar case, it was explicitly checked that all edges are present in G and thus it is valid. On the other hand, if no edge $e \in E_K$ of the Kuratowski subdivision is part of a triangle, then by Lemma 3.1 G cannot be a disk graph.

Running Time The running time of the planarity test of Boyer and Myrvold, as well as the running time for finding the girth in a planar graph are in $O(n)$ [1,2]. Computing the sorted adjacency lists takes $O(n + m)$ time by Lemma 2.1. Each vertex of G_K has degree at most four. Thus, the adjacency list of each vertex is considered $O(1)$ times, when explicitly checking all edges in E_K. The procedure described above for finding a common vertex in two adjacency lists takes time proportional to their length. Hence, the overall time spent on traversing adjacency lists is $O(m)$. The stated running time of $O(n + m)$ for our algorithm follows. ◀

The robust algorithm for computing the girth in the domain of general disk graphs directly follows from Theorem 3.2. In the planar case, we compute the girth as before but return the value even if it is more than three. In the non-planar case, there has to be a triangle and the girth is 3 or the input graph is not a disk graph. Thus, we do the same as in Theorem 3.2 except for when we find a triangle, we return 3 and not the triangle itself.

▶ **Corollary 3.3.** *There is a robust algorithm to compute the girth of a graph in the domain of general disk graphs that runs in $O(n + m)$ time.*

4 Conclusion

We showed that a clever use of Kuratowski subdivisions leads to a robust linear time algorithm for finding triangles and computing the girth in the domain of general disk graph. It would be interesting to see if there are properties of general disk graphs and transmission graphs that allow sublinear running times similar to the unit disk graph algorithm by Klost and Mulzer [7]. We believe that this is just the beginning of a systematic study of efficient robust algorithms for abstract representations of geometric intersection graphs. For example, we think that our results can be generalized to intersection graphs of more general objects, such as pseudodisks or homothets of a single fat convex object.

References

1 John Boyer and Wendy Myrvold. On the Cutting Edge: Simplified O(n) Planarity by Edge Addition. 8(3):241–273. URL: `https://jgaa.info/index.php/jgaa/article/view/paper91`, `doi:10.7155/jgaa.00091`.

2 Hsien-Chih Chang and Hsueh-I Lu. Computing the girth of a planar graph in linear time. *SIAM J. Comput.*, 42(3):1077–1094, 2013. `doi:10.1137/110832033`.

3 William Evans, Mereke van Garderen, Maarten Löffler, and Valentin Polishchuk. Recognizing a DOG is hard, but not when it is thin and unit. In *Proc. 8th FUN*, pages 16:1–16:12, 2016. `doi:10.4230/LIPIcs.FUN.2016.16`.

4 John Hopcroft and Robert Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974. doi:10.1145/321850.321852.

5 Ross J. Kang and Tobias Müller. Sphere and dot product representations of graphs. *Discrete Comput. Geom.*, 47(3):548–568, 2012. doi:10.1007/s00454-012-9394-8.

6 Haim Kaplan, Katharina Klost, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Triangles and girth in disk graphs and transmission graphs. In *Proc. 27th Annu. European Sympos. Algorithms (ESA)*, pages 64:1–64:14, 2019. Full version available at https://arxiv.org/abs/1907.01980. doi:10.4230/LIPIcs.ESA.2019.64.

7 Katharina Klost and Wolfgang Mulzer. Robust Algorithms for Finding Triangles and Computing the Girth in Unit Disk and Transmission Graphs. URL: http://arxiv.org/abs/2405.01180, arXiv:2405.01180, doi:10.48550/arXiv.2405.01180.

8 Casimir Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, 15(1):271–283, 1930. URL: http://eudml.org/doc/212352.

9 Vijay Raghavan and Jeremy Spinrad. Robust algorithms for restricted domains. *J. Algorithms*, 48(1):160–172, 2003. doi:10.1016/S0196-6774(03)00048-8.

Improved Bound on the Number of Pseudoline Arrangements via the Zone Theorem

Justin Dallant[1]

1 Department of Computer Science, Aarhus University, Denmark
justindallant@cs.au.dk

Abstract

Pseudoline arrangements are fundamental objects in discrete and computational geometry, and different works have tackled the problem of improving the known bounds on the number of simple arrangements of n pseudolines over the past decades. The lower bound in particular has seen two successive improvements in recent years (Dumitrescu and Mandal in 2020 and Cortés Kühnast et al. in 2024). Here we focus on the upper bound, and show that for large enough n, there are at most $2^{0.6496n^2}$ different simple arrangements of n pseudolines. This follows a series of incremental improvements starting with work by Knuth in 1992 showing a bound of roughly $2^{0.7925n^2}$, then a bound of $2^{0.6975n^2}$ by Felsner in 1997, and finally the previous best known bound of $2^{0.6572n^2}$ by Felsner and Valtr in 2011. The improved bound presented here follows from a simple argument to combine the approach of this latter work with the use of the Zone Theorem.

1 Introduction

An arrangement of pseudolines in the Euclidean plane is a finite set of n simple curves extending to infinity in both directions such that every two curves intersect at exactly one point where they cross. They have been the subject of numerous works, the earliest of which going back to at least the 1920's [13]. We refer the reader to the corresponding chapter of the Handbook [15, Chapt. 5] for a thorough overview of the subject.

We will always assume pseudoline arrangements are *simple*, meaning that no three pseudolines intersect at a common point. Following [6], we will consider our arrangements to be *marked*[1], meaning that they come with a distinguished unbounded cell called the *north-cell* (the unique cell lying on the other side of all pseudolines is called the *south-cell*). Two pseudoline arrangements are *isomorphic* if one can be mapped to the other one by an orientation preserving homeomorphism of the plane which preserves the north-cell.

Wiring diagrams. One particularly nice way of drawing (in our case simple) pseudoline arrangements, introduced by Goodman [8] is through *wiring diagrams.* In a wiring diagram, the n pseudolines are constrained to lie on n horizontal lines (or "wires"), except around points where they cross another pseudoline and move to a neighboring wire. Figure 1 shows an example of an arbitrarily drawn pseudoline arrangement next to a realization of the same arrangement as a wiring diagram.

Number of pseudoline arrangements. It is known that the number B_n of non-isomorphic arrangements of n pseudolines is $2^{\Theta(n^2)}$ but the leading hidden constant in the exponent is not known. Let $c^- = \liminf_{n\to\infty} \frac{\lg B_n}{n^2}$ and $c^+ = \limsup_{n\to\infty} \frac{\lg B_n}{n^2}$ (throughout the paper,

[1] Note that the final bound we obtain here applies similarly to unmarked arrangements or arrangements in the projective plane, as these differences only affect lower order factors.

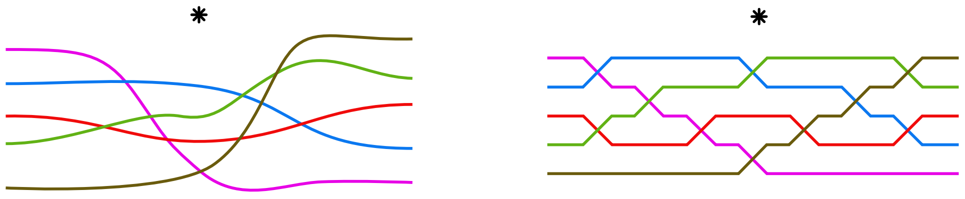


Figure 1 An arbitrarily drawn pseudoline arrangement (left) and the same arrangement drawn as a wiring diagram (right). In both cases, the north-cell is marked by a star.

we use lg to denote the logarithm in base 2). It is in fact an open question whether $c^- = c^+$, i.e. whether the limit of $\frac{\lg B_n}{n^2}$ exists. The first bound on c^- was given by Goodman and Pollak in the 1980's [9] who showed $c^- \geq 1/8$. This was improved in the 1990's to $c^- \geq 1/6$ by Knuth [10], who also showed $c^+ < 0.79249$ and conjectured $c^+ \leq 0.5$. The upper bound was then improved by Felsner [5] to $c^+ < 0.6975$ and finally to $c^+ < 0.6572$ in 2011 by Felsner and Valtr [6] who also showed $c^- \geq 0.1887$. The lower bound saw some recent additional improvements. In 2020, Dumitrescu and Mandal [4] showed $c^- > 0.2083$. In 2024, Cortés Kühnast, Felsner and Scheucher [12] and Dallant [2] independently discovered similar methods to improve the previous construction, which resulted in a merged paper [11] showing the currently best known bound of $c^- > 0.2721$.

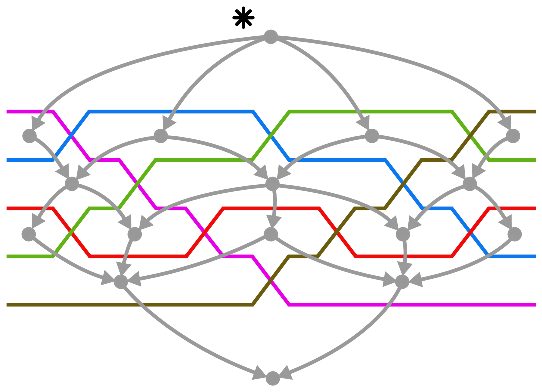


Figure 2 A pseudoline arrangement and its associated directed acyclic dual graph.

Cutpaths and upper bound. Consider the directed acyclic graph which is dual to the wiring diagram, with edges oriented from north to south (see Figure 2). A path from the north-cell to the south-cell in the graph is called a *cutpath* of the arrangement. Informally, the set of all cutpaths represents all combinatorially distinct ways one can insert a new pseudoline in the pseudoline arrangement. The approach of Felsner and Valtr for the upper bound, as did that of Knuth earlier, relies on bounding the quantity γ_n, defined as the maximum number of cutpaths an arrangement of n pseudolines can have. As shared in a remark by Knuth [10, p. 97], an argument by Bern shows that one can use the sharpest version of the Zone Theorem to show that $\gamma_n \leq O(2.711^n)$, yielding the bound $c^+ < 0.7194$. Knuth also conjectured that the "bubblesort arrangements" of size n, which (in a slightly different setting than the one presented here) have $n2^{n-2}$ cutpaths, maximize this number.[2] If true, this would yield $c^+ \leq 0.5$. Felsner and Valtr show $\gamma_n \leq O(2.487^n)$ without using the Zone Theorem, resulting in the previously best known bound of $c^+ < 0.6572$. They also report that a construction by Ondřej Bílka shows $\gamma_n \geq \Omega(2.076^n)$, thus disproving Knuth's conjecture on the maximum number of cutpaths.

We use a simple argument to combine their method with the use of the Zone Theorem, thus improving the upper bound on the number of cutpaths to roughly $\gamma_n \leq O(2.461^n)$, in turn resulting in $c^+ < 0.6496$. The argument, given in the next section, results in a bound expressed as the maximum of a certain function over some domain. We then explain how to obtain an upper bound on this function through a branch and bound through linear relaxation based approach, resulting in our final bound. We also describe in the full version of this paper [3] a family of arrangements showing $\gamma_n \geq \Omega(2.083^n)$, a slight improvement on the construction by Bílka.

2 Improved upper bound on the number of pseudoline arrangements

We start by giving a brief overview of the approach of Felsner and Valtr [6] to bound the maximum possible number γ_n of cutpaths in an arrangement of n pseudolines. Using the fact that $B_{n+1} \leq \gamma_n \cdot B_n$, this then yields a bound on B_n by induction.

Consider an arrangement A of n pseudolines represented as a wiring diagram, together with a cutpath p traversing A from the north cell to the south cell. The path of p can be described by choosing for each cell it reaches, which pseudoline bounding the cell from below it crosses to exit the cell. We classify these exits into different categories. Consider a cell f with d exits (i.e. d pseudolines bounding it from below), which we label $1, 2, \ldots, d$ from left to right. If $d = 1$, we call the single existing exit the *unique exit* of f. Otherwise, we call exit 1 the *left exit* of f and exit d the *right exit* of f. All other exits are called *middle exits* of f.

The following fact was proven by Knuth.

▶ **Lemma 2.1** ([10]). *Any pseudoline in A can appear at most once as a middle exit of a cell visited by p.*

If the cutpath crosses k pseudolines as a middle exit and u as a unique exit, one can thus ecode it by choosing the k pseudolines the cutpath will cross as a middle exit, and fixing some binary string β_p of length $n - k - u$. Any time a cell is reached, we check if it has a unique exit or if one of the middle exits is among the k chosen, and we take that exit if this is the case. If it is not the case (which happens at most $n - k - u$ times), we consume one entry of β_p to choose whether we take the left or right exit. This encoding shows that for any fixed k, the number of cutpaths taking k middle exits and u unique exits is at most $\binom{n}{k}2^{n-k-u}$.

This bound can be further refined by analyzing the repartition of middle exits among visited cells more carefully, showing that the number of cutpaths taking k middle exits and u unique exits is at most

$$\binom{n-u}{k}\left(\frac{n}{n-u}\right)^k 2^{n-k-u}.$$

[2] Knuth verified this to be true for $n \leq 9$. However, in the setting of the present paper, this is not true even for small n, where the "odd-even sort arrangements" maximize this number at least up to $n = 10$ (note that in Knuth's setting, these also have $n2^{n-2}$ cutpaths). Thanks to Günter Rote for verifying this.

Here we have encoded the cutpath by describing its course from the north-cell to the south-cell, but one can also choose to encode it in the other direction, by rotating the whole plane by 180 degrees and describing its course from the south-cell to the north-cell. The crucial observation made by Felsner and Valtr is that any middle exit in the original description corresponds to a unique exit in the rotated description (and similarly any middle exit in the rotated description corresponds to a unique exit in the original description). Felsner and Valtr exploit the refined bound, together with the freedom to choose in which direction to encode a cutpath, to show their best bound of roughly $\gamma_n \leq 4n \cdot 2.487^n$.

Improving the upper bound.

In order to improve the upper bound, we will pay closer attention to the total number of middle exits among all cells visited by p. Let $\Gamma(m, k, u, m', k', u')$ denote the set of cutpaths of A such that when viewed from the north-cell to the south-cell

- the total number of middle exits taken is k,
- the total number of unique exits taken is u,
- the total number of middle exits among cells visited (including those which were not taken) is m,

and when viewed from the south-cell to the north-cell

- the total number of middle exits taken is k',
- the total number of unique exits taken is u',
- the total number of middle exits among cells visited (including those which were not taken) is m',

Stated more precisely, the bound shown by Felsner and Valtr is the following[3].

▶ **Lemma 2.2** ([6]). *For all m, k, u, m', k', u' such that $k \leq m$ and $k \leq n - u$, we have*

$$|\Gamma(m, k, u, m', k', u')| \leq \binom{n-u}{k} \left(\frac{m}{n-u}\right)^k 2^{n-k-u}.$$

By symmetry we also have that for $k' \leq m'$ and $k' \leq n - u'$,

$$|\Gamma(m, k, u, m', k', u')| \leq \binom{n-u'}{k'} \left(\frac{m'}{n-u'}\right)^{k'} 2^{n-k'-u'}.$$

Note that the conditions $k \leq m$ and $k \leq n-u$ (and similarly for $k' \leq m'$ and $k' \leq n-u'$) always hold, as a cutpath cannot take more middle exits than there are such exits, and a taken exit cannot be both a middle and a unique exit simultaneously. The correspondence between middle exits and unique exits in the rotated arrangement implies $k \leq u'$ and $k' \leq u$. Lemma 2.1 implies $m \leq n$ and $m' \leq n$.

In order to get a tighter grip on m and m', our simple idea is to complement the use of Lemma 2.1 to bound them separately with the use of (the tight version of) the Zone Theorem to bound them conjointly (one could obtain the same bound we obtain here without using Lemma 2.1 at all, but its use makes some of the arguments simpler). Call *zone* of a pseudoline

[3] Felsner and Valtr state the bound only for the case $n - u \leq m$. However, their proof easily carries over to the case $n - u > m$ by replacing the partitions of $[n]$ into $n - u$ subsets they consider in their proof with colorings of $[n]$ using $n - u$ colors. Alternatively, one can notice that the almost immediate upper bound of $\binom{m}{k} 2^{n-u-k}$ is less than the bound stated in the lemma when $n - u > m$, as shown in the full version of this paper [3].

the set of cells supported by that pseudoline, and call *complexity of the zone* the sum of the number of edges over all the cells in the zone (an edge may be counted twice if it appears in two different cells of the zone). Then the following holds.

▶ **Theorem 2.3** (Zone Theorem [1, 14])**.** *In an arrangement of $n+1$ pseudolines, the complexity of the zone of any given pseudoline is at most $9.5n-3$.*

In the full version of this paper [3], we use these constraints and a procedure based on linear programming and branch-and-bound, to show that $\gamma_n \leq n^6 \cdot 2^{\alpha \cdot n}$ where $\alpha < 1.2992$. This together with $B_{n+1} \leq \gamma_n B_n$ yields our final bound.

▶ **Theorem 2.4.** *For large enough n, the number B_n of simple arrangements of n pseudolines is at most $2^{0.6496n^2}$.*

Proof. Let n_0 be an integer such that for all $n \geq n_0$, $\gamma_n \leq 2^{\alpha \cdot n}$, where $\alpha < 1.2992$. For any $n \geq n_0$, we thus have $B_{n+1} \leq 2^{\alpha \cdot n} B_n$. By induction, it follows that

$$B_{n+1} \leq 2^{\alpha \cdot (n_0+(n_0+1)+\ldots n)} B_{n_0} \leq 2^{\alpha \frac{(n-n_0+1)(n_0+n)}{2}} B_{n_0}.$$

As $\alpha < 1.2992$ this last expression is at most $2^{0.6496n^2}$ for large enough n. ◀

In the full version [3] we also provide a construction showing the following.

▶ **Theorem 2.5.** *For any $n > 0$, there exists an arrangement of n pseudolines with $\Omega(2.083^n)$ cutpaths.*

3 A potential avenue for improvements

In our upper bound on B_n, we have exploited the leading constant of 9.5 in the tight Zone Theorem. While this constant can not be improved in general, we note that we do not need the full power of this theorem in our proof. Indeed, by appropriately choosing the order in which we encode the pseudolines, it is enough for our purpose to know that in any pseudoline arrangement, at least one of the lines has a small zone complexity (as opposed to all of the lines having at most a certain zone complexity). This would lead to an additional term of order $\Theta(n \lg n)$ in the length (in bits) of the encoding, but this is negligible in front of the leading term of order $\Theta(n^2)$. In this light, we propose the two following conjectures (the second being a weaker consequence of the first).

▶ **Conjecture 1.** *Let $\mathcal{L}$ be an arrangement of n pseudolines. Then the average of the zone complexities of the pseudolines in $\mathcal{L}$ is at most $9n + O(1)$.*

▶ **Conjecture 2.** *Let $\mathcal{L}$ be an arrangement of n pseudolines. Then there is at least one pseudoline in $\mathcal{L}$ whose zone complexity is at most $9n + O(1)$.*

The question of improving the average zone complexity was previously raised by Zerbib [16] (without proposing an explicit constant), who showed that a weaker statement implied by such an improvement holds true. Note that we cannot hope for a better constant than 9 in our conjectures, as one can construct even straight line arrangements meeting this bound (for example by taking all lines supporting the edges of a regular convex $(2k+1)$-gon). If either conjecture holds true, our method would give an upper bound of $B_n < 2^{0.6074n^2}$ for large enough n. We note in passing that a confirmation of Conjecture 1 could also for example replace the use of the Zone Theorem in the work of Goaoc and Welzl [7]. This would improve the upper bound on the variance of the number of extreme points of a uniformly random labeled n-point order type from 3 to $2 + o(1)$.

References

1 Marshall W Bern, David Eppstein, Paul E Plassmann, and F Frances Yao. Horizon theorems for lines and polygons. *Discrete and Computational Geometry*, 6:45–66, 1990.

2 Justin Dallant. Improved lower bound on the number of pseudoline arrangements. *arXiv preprint arXiv:2402.13923*, 2024.

3 Justin Dallant. Improved bound on the number of pseudoline arrangements via the zone theorem. *arXiv preprint arXiv:2502.20909*, 2025.

4 Adrian Dumitrescu and Ritankar Mandal. New lower bounds for the number of pseudoline arrangements. *Journal of Computational Geometry*, 11(1):60–92, 2020.

5 Stefan Felsner. On the number of arrangements of pseudolines. In *Proceedings of the twelfth annual Symposium on Computational Geometry*, pages 30–37, 1996.

6 Stefan Felsner and Pavel Valtr. Coding and counting arrangements of pseudolines. *Discrete & Computational Geometry*, 46(3):405–416, 2011.

7 Xavier Goaoc and Emo Welzl. Convex hulls of random order types. *Journal of the ACM*, 70(1):1–47, 2023.

8 Jacob E. Goodman. Proof of a conjecture of Burr, Grünbaum, and Sloane. *Discrete Mathematics*, 32(1):27–35, 1980.

9 Jacob E. Goodman and Richard Pollack. On the combinatorial classification of nondegenerate configurations in the plane. *Journal of Combinatorial Theory, Series A*, 29(2):220–235, 1980.

10 Donald E. Knuth. *Axioms and hulls.* Lecture Notes in Computer Science. Springer, 1992.

11 Fernando Cortés Kühnast, Justin Dallant, Stefan Felsner, and Manfred Scheucher. An improved lower bound on the number of pseudoline arrangements. In *40th International Symposium on Computational Geometry, SoCG 2024, Athens, Greece*, pages 43:1–43:18, 2024.

12 Fernando Cortés Kühnast, Stefan Felsner, and Manfred Scheucher. An improved lower bound on the number of pseudoline arrangements. *arXiv preprint arXiv:2402.13107*, 2024.

13 Friedrich Levi. Die Teilung der projektiven Ebene durch Gerade oder Pseudogerade. *Ber. Math.-Phys. Kl. Sächs. Akad. Wiss*, 78:256–267, 1926.

14 Rom Pinchasi. The zone theorem revisited. *Manuscript*, 2011. URL: `http://www2.math.technion.ac.il/~room/ps_files/zonespl.pdf`.

15 Csaba D. Tóth, Joseph O'Rourke, and Jacob E. Goodman. *Handbook of discrete and computational geometry.* CRC press, 2017.

16 Shira Zerbib. On the zone complexity of a vertex. *SIAM Journal on Discrete Mathematics*, 25(2):719–730, 2011.

A fractional Helly theorem for set systems with slowly growing homological shatter function

Marguerite Bin[1]

1 Université de Lorraine, CNRS, INRIA, LORIA, F-54000 Nancy, France.
marguerite.bin@loria.fr

Abstract

We study the fractional Helly number of families of sets in $\mathbb{R}^d$ where every intersection between t members of the family has its Betti numbers bounded from above by a function of t. Although the Radon number of such families may not be bounded, we show that their fractional Helly number is $d+1$. To achieve this, we introduce graded analogues of the Radon and Helly numbers. This generalizes previously known fractional Helly theorems.

Related Version arXiv:2411.18605

1 Introduction

The well-known theorem of Helly [3] states that the intersection of all members of a finite family of convex sets in $\mathbb{R}^d$ is nonempty if and only if every $d+1$ members of the family intersect. We say that the family of all convex sets in $\mathbb{R}^d$ has *Helly number* $d+1$. Here, we are interested in yet another parameter that arises when studying families of convex sets: the *fractional Helly number* of an infinite family $\mathcal{F}$. It is the smallest integer s such that for every finite subfamily $\mathcal{F}' \subset \mathcal{F}$, if an α-fraction of the s-tuples of $\mathcal{F}'$ intersect, then some $\beta_{\mathcal{F}}(\alpha)|\mathcal{F}'|$ members of $\mathcal{F}'$ intersect. In the case of the family of convex sets in $\mathbb{R}^d$, it is $d+1$ [7]. The fractional Helly number seems more robust than the Helly number. For instance, Matoušek proved [9] that the family $\mathcal{A}_{B,d} := \{P \leq 0\}_{P\in\mathbb{R}_B[X_1,\dots,X_d]}$ has a fractional Helly number of $d+1$, even though the Helly number of this family is unbounded.

Kalai and Meshulam conjectured a fractional Helly theorem that generalizes Matoušek's result. Before presenting it, we introduce the (hth) *homological shatter function* ([6], see [1]), which describes the maximal topological complexity of intersections among increasingly many sets. It is defined as

$$\phi^{(h)}_{\mathcal{F}} : \begin{cases} \mathbb{N} & \to \quad \mathbb{N}\cup\{\infty\} \\ k & \mapsto \quad \sup\left\{\tilde{\beta}_i\left(\bigcap_{F\in\mathcal{G}} F, \mathbb{Z}_2\right) \mid \mathcal{G}\subset\mathcal{F}, |\mathcal{G}|\leq k, 0\leq i\leq h\right\} \end{cases},$$

where $\tilde{\beta}_i(X,\mathbb{Z}_2)$ denotes the i-th reduced Betti number with coefficients in $\mathbb{Z}_2$ of the topological space X. We can now state the following combination of two conjectures by Kalai and Meshulam ([6, Conjectures 6 and 7], see [1, Conjecture 1.9 and 1.10]):

▶ **Conjecture 1.1** (Kalai and Meshulam, [6]). *For every (possibly infinite) family $\mathcal{F}$ of sets in $\mathbb{R}^d$ with $\phi^{(d)}_{\mathcal{F}}$ growing at most polynomially, the fractional Helly number of $\mathcal{F}$ is at most $d+1$.*

Notice that this conjecture implies Matoušek's result. Indeed, the Milnor-Thom theorem [10, Theorem 3] states that $\phi^{(h)}_{\mathcal{A}_{B,d}}(k) \leq (1+Bk)^d$.

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Goaoc, Holmsen, and Patáková [1, Corollary 1.3] proved a weaker version of the conjecture, where the ($\lceil d/2 \rceil$th) homological shatter function $\phi_{\mathcal{F}}^{(\lceil d/2 \rceil)}$ is bounded by a constant instead of being a polynomial. In this case, we can define the *(hth) homological complexity* of $\mathcal{F}$ as:

$$\mathrm{HC}_h(\mathcal{F}) := \sup_{k \in \mathbb{N}} \phi_{\mathcal{F}}^{(h)}(k).$$

Contributions

Here, we prove that results used by Goaoc et al. [1] can be combined to prove a new theorem that brings us a step closer to Conjecture 1.1.

▶ **Theorem 1.2.** *For every integers $b, d \geq 0$ and $\alpha \in (0,1)$, there exists a nonstationary function $\Psi_{d,b} : \mathbb{N} \to \mathbb{N}$ such that: If $\mathcal{F}$ is a family of subsets of $\mathbb{R}^d$ satisfying $\phi_{\mathcal{F}}^{(\lceil \frac{d}{2} \rceil)} \leq \Psi_{d,b}$, then $\mathcal{F}$ has a fractional Helly number of at most $d+1$.*

Unfortunately, despite being nonstationary, the functions $\Psi_{b,d}$ are far from having a polynomial growth; they are somewhere between the inverse Ackermann function and the iterated logarithm (log^*).

A convenient method for adapting the proof of Goaoc et al. is to introduce graded analogues of the standard parameters of $\mathcal{F}$ as a convexity space. This new framework enables us to directly apply their results rather than transposing each proof. These graded parameters may also be of independent interest.

2 Standard parameters

In order to prove Theorem 1.2, let us recall the main results used by Goaoc et al. We first need to define the notion of $\mathcal{F}$-convex hull. Given a family $\mathcal{F}$ on a ground set X, we define the *$\mathcal{F}$-convex hull* of a set $P \subset X$, denoted by $\mathsf{conv}_{\mathcal{F}}(P)$, as the intersection between all the members of $\mathcal{F}$ containing P. In particular, this allows us to define four parameters.

- The *Radon number* $\mathrm{r}(\mathcal{F})$ of a family $\mathcal{F}$, denoted by $\mathrm{r}(\mathcal{F})$, is the smallest integer r such that every set $S \subset X$ of cardinality r can be split into two nonempty disjoint parts $S = P_1 \sqcup P_2$ satisfying $\mathsf{conv}_{\mathcal{F}}(P_1) \cap \mathsf{conv}_{\mathcal{F}}(P_2) \neq \emptyset$.
- The *Helly number* of a family $\mathcal{F}$, denoted by $\mathrm{h}(\mathcal{F})$, is the smallest integer h with the following property: If in a finite subfamily $S \subset \mathcal{F}$, every $h' \leq h$ members of S intersect (we say that S forms a *h-clique*), then all members of S intersect. If no such h exists, we set $\mathrm{h}(\mathcal{F}) = \infty$.
- The k-th *colorful Helly number* of a family $\mathcal{F}$, denoted by $\mathrm{ch}_k(\mathcal{F})$, is the smallest number of colors $m \geq k$ such that for every coloring of a subfamily $\mathcal{F}' \subset \mathcal{F}$ with m colors, if every subfamily that contains one element of each color forms a k-clique, then at least one color class itself forms a k-clique.[1]
- The *k-th fractional Helly number* of a family $\mathcal{F}$, denoted by $\mathrm{fh}_k(\mathcal{F})$ is the smallest integer s such that there exists a function $\beta_{\mathcal{F}} : (0,1) \to (0,1)$ with the following property: For every finite subfamily $\mathcal{F}' \subset \mathcal{F}$, whenever a fraction α of the s-tuples of $\mathcal{F}'$ forms a k-clique, a subset $\mathcal{G}$ of $\mathcal{F}'$ of size $\beta_{\mathcal{F}}(\alpha)|\mathcal{F}'|$ forms a k-clique. When $k = \infty$, a k-clique is a subset where the intersection among all its elements is nonempty: $\mathrm{fh}(\mathcal{F}) := \mathrm{fh}_\infty(\mathcal{F})$ defines the regular *fractional Helly number.*

[1] When $k = \infty$, it does not define the regular colorful Helly number. However, when $k = \mathrm{h}(\mathcal{F})$, it does.

Notice that when $\mathrm{h}(\mathcal{F}) \leq k$, $\mathrm{fh}_k(\mathcal{F})$ coincides with the fractional Helly number of $\mathcal{F}$.

Several relations hold between such parameters for every family[2]. Levi's inequality [8] states that $\mathrm{h}(\mathcal{F}) \leq \mathrm{r}(\mathcal{F}) - 1$. More recently, Holmsen and Lee [4, Theorem 1.2] proved that the k-th fractional Helly number is bounded from above by the k-th colorful Helly number, which in turn is bounded from above by a function $m(\cdot)$ of the Radon number [5, Lemma 2.3]:

$$\mathrm{fh}_k(\mathcal{F}) \leq \mathrm{ch}_k(\mathcal{F}) \qquad \text{for any } k \in \mathbb{N}, \tag{1}$$

$$\mathrm{ch}_k(\mathcal{F}) \leq \max\left(m\left(\mathrm{r}(\mathcal{F})\right), k\right) \quad \text{if } \mathrm{h}(\mathcal{F}) \leq k, \text{ with } m(\cdot) \text{ defined by [5, Lemma 2.3]}, \tag{2}$$

When the ground set is $\mathbb{R}^d$, Goaoc et al. [2] first proved that a function of the homological complexity and the dimension bounds the Helly number from above. Patáková [11] later improved the result to prove that it also bounds the Radon number from above:

$$\mathrm{h}(\mathcal{F}) < \mathrm{r}(\mathcal{F}) \leq r\left(\mathrm{HC}_{(\lceil \frac{d}{2} \rceil)}(\mathcal{F}), d\right) \qquad \text{with } r(\cdot,\cdot) \text{ defined by [11, Theorem 2.1]}. \tag{3}$$

Combined with Inequalities (2) and (1), it implies that a finite homological complexity ensures a finite fractional Helly number [11, Theorem 2.3]. Goaoc et al. [1, Corollary 1.3] later improved the result; a finite homological complexity actually ensures a fractional Helly number of at most $d+1$.

3 Graded parameters

The homological shatter function can be viewed as a graded analogue of the homological complexity, in the sense that it is equal to:

$$\phi_{\mathcal{F}}^{(h)}(t) = \sup_{\substack{\mathcal{F}' \subset \mathcal{F} \\ |\mathcal{F}'| = t}} \mathrm{HC}_h(\mathcal{F}').$$

We can similarly define the t-th graded Radon, Helly, and k-colorful Helly numbers of the family $\mathcal{F}$ as:

$$\mathrm{r}^{(t)}(\mathcal{F}) \coloneqq \sup_{\substack{\mathcal{F}' \subset \mathcal{F} \\ |\mathcal{F}'| \leq t}} \mathrm{r}(\mathcal{F}'), \quad \mathrm{h}^{(t)}(\mathcal{F}) \coloneqq \sup_{\substack{\mathcal{F}' \subset \mathcal{F} \\ |\mathcal{F}'| \leq t}} \mathrm{h}(\mathcal{F}'), \quad \mathrm{ch}_k^{(t)}(\mathcal{F}) \coloneqq \sup_{\substack{\mathcal{F}' \subset \mathcal{F} \\ |\mathcal{F}'| \leq t}} \mathrm{ch}_k(\mathcal{F}').$$

3.1 Relations in the graded realm

These graded numbers follow the same relations as their ungraded analogues; we can consider all subfamilies of size t and apply the known relations between the ungraded numbers. We get the following relation by applying Inequality (2):

$$\mathrm{ch}_k^{(t)}(\mathcal{F}) \leq \max\left(m\left(\mathrm{r}^{(t)}(\mathcal{F})\right), k\right) \qquad \text{if } \mathrm{h}^{(t)}(\mathcal{F}) \leq k. \tag{4}$$

When the ground set of $\mathcal{F}$ is $\mathbb{R}^d$, applying Inequality (3) also yields:

$$\mathrm{h}^{(t)}(\mathcal{F}) < \mathrm{r}^{(t)}(\mathcal{F}) \leq r\left(\phi_{\mathcal{F}}^{(\lceil \frac{d}{2} \rceil)}(t), d\right). \tag{5}$$

[2] The relations were initially proved for convexity spaces, but Patáková [11] mentioned they also hold for families of sets. We can also adapt the definitions of graded parameters to consider only parameters of convexity spaces.

Naively transposing Inequality (1) using the graded formalism does not seem relevant. However, upon closer inspection of Holmsen's result [4, Theorem 1.2], it turns out that it states a more general inequality, providing the first bridge between the graded and ungraded parameters:

$$\mathrm{fh}_k(\mathcal{F}) \leq \mathrm{ch}_k^{(k\ell)}(\mathcal{F}) \qquad\qquad \text{if } \mathrm{ch}_k^{(k\ell)}(\mathcal{F}) \leq \ell \text{ and } k < \ell. \tag{6}$$

Indeed, for every integers $k < \ell$, having $\ell' := \mathrm{ch}_k^{(k\ell)}(\mathcal{F}) \leq \ell$ means that some patterns of fixed size $k\ell'$ (called the complete ℓ'-tuples of missing edges [5, §3]) are forbidden in the hypergraph of k-intersections. This is the only property needed to ensure that $\mathrm{fh}_k(\mathcal{F}) \leq \ell'$.

If we want to end up with a fractional Helly number rather than just a k-fractional Helly number, we need the parameter k to be greater than $\mathrm{h}(\mathcal{F})$; in particular, we need $\mathrm{h}(\mathcal{F})$ to be finite. Fortunately, we can relate the growth of the graded Helly numbers to the Helly number: Graded Helly numbers must either grow fast or be stationary.

▶ **Lemma 3.1.** *If* $\mathrm{h}^{(t)}(\mathcal{F}) < t$ *for all* $t > t_0$*, then* $\mathrm{h}(\mathcal{F}) \leq t_0$*.*

Proof. Notice that we have $\mathrm{h}^{(t)}(\mathcal{F}) > \mathrm{h}^{(t-1)}(\mathcal{F})$ if and only if $\mathrm{h}^{(t)}(\mathcal{F}) = t$. Thus, the assumption implies that the sequence $\mathrm{h}^{(t)}(\mathcal{F})$ is stationary starting from t_0. Since $\mathrm{h}^{(t)}(\mathcal{F}) \xrightarrow[t\to\infty]{} \mathrm{h}(\mathcal{F})$, it follows that $\mathrm{h}(\mathcal{F}) = \mathrm{h}^{(t_0)}(\mathcal{F}) \leq t_0$. ◀

3.2 Wrapping up the relations

We have all the analogues of the relations needed for the proof. We first combine Inequality (5) with Inequality (4):

$$\mathrm{ch}_k^{(t)}(\mathcal{F}) \leq m\left(r\left(\phi_{\mathcal{F}}^{(\lceil \frac{d}{2}\rceil)}(t), d\right)\right) \qquad \text{if } \mathrm{h}^{(t)}(\mathcal{F}) \leq k \leq m\left(r\left(\phi_{\mathcal{F}}^{(\lceil \frac{d}{2}\rceil)}(t), d\right)\right). \tag{7}$$

We can then combine Inequality (6) with Inequality (7): If there exist k, ℓ such that

$$m\left(r\left(\phi_{\mathcal{F}}^{(\lceil \frac{d}{2}\rceil)}(k\ell), d\right)\right) \leq \ell \quad \text{and} \quad \mathrm{h}^{(k\ell)}(\mathcal{F}) \leq k < \ell, \tag{8}$$

then $\mathrm{fh}_k(\mathcal{F}) \leq m\left(r\left(\phi_{\mathcal{F}}^{(\lceil \frac{d}{2}\rceil)}(k\ell), d\right)\right)$.

We now relate the Helly number with the homological shatter function. We can check that, by combining Equation (5) and Lemma 3.1, for any $b_0 \geq 0$, if the family satisfies

$$\forall b' \geq b_0, \quad \phi_{\mathcal{F}}^{(\lceil \frac{d}{2}\rceil)}(r(b'+1, d)) \leq b', \tag{9}$$

then $\mathrm{h}(\mathcal{F}) < r(b_0, d)$. Indeed, it ensures that for any $t \in \{r(b', d), \ldots, r(b'+1, d)\}$, $r^{(t)}(\mathcal{F}) \leq t$ since r and $\phi_{\mathcal{F}}^{(\lceil \frac{d}{2}\rceil)}$ are non-decreasing.

Let us assume that the condition (9) is satisfied for some b_0. It remains to find the right values for k, ℓ that satisfy Condition (8). As previously mentioned, we want to set $k \geq \mathrm{h}(\mathcal{F})$, say $k = r(b_0, d)$. Satisfying Condition (8) comes down to looking for a value $\ell > k$ such that:

$$m\left(r\left(\phi_{\mathcal{F}}^{(\lceil \frac{d}{2}\rceil)}(\ell r(b_0, d)), d\right)\right) \leq \ell. \tag{10}$$

3.3 Definition of the function Ψ_{d,b_0}

To summarize conditions (9) and (10), we introduce a nonstationary function $\Psi_{d,b_0} : \mathbb{N} \to \mathbb{N}$ that satisfies Condition (9) and $\Psi_{d,b_0}(m(r(b_0,d))r(b_0,d)) \leq b_0$. We can check that any function bounded from above by such a Ψ_{d,b_0} satisfies Conditions (9) and (10) with $\ell = m(r(b_0,d))$ (since the functions m and r are non-decreasing).

To make such a function explicit, we inverse $R_d : b' \mapsto r(b'+1,d)$ by defining $S : \mathbb{N} \to \mathbb{N}$ as $S(x) = \max\{b' \in \mathbb{N} \mid x \geq r(b'+1,d)\}$. Next, we define $\Psi_{d,b_0} : \mathbb{N} \to \mathbb{N}$ as follows:

$$\Psi_{d,b_0}(t) = \begin{cases} b_0 & \text{if } \; r(b_0,d) \leq t \leq m(r(b_0,d))r(b_0,d) \\ S(t) & \text{if } \; t > m(r(b_0,d))r(b_0,d) \end{cases}.$$

Note that the function S is defined as the inverse of the rapidly growing function R_d. Patáková [11] does not specify an upper bound, but the function r seems to be (very roughly) bounded from above by a function in $\mathcal{E}^5$ in the Grzegorczyk hierarchy. Consequently, the best growth we can currently hope for S, and thus for $\Psi_{d,b}$, is only marginal: S is somewhere between the inverse Ackermann function and the iterated logarithm (log^*).

4 Proof of the theorem

We can finally prove Theorem 1.2.

Proof. Let $\mathcal{F}$ a family of subsets of $\mathbb{R}^d$ satisfying $\phi_{\mathcal{F}}^{(\lceil \frac{d}{2} \rceil)} \leq \Psi_{d,b_0}$.

- The function $\phi_{\mathcal{F}}^{(\lceil \frac{d}{2} \rceil)}$ satisfies Condition (10) for $k = r(b_0,d)$ and $\ell = m(r(b_0,d))$, ensuring that $\mathrm{fh}_k(\mathcal{F}) \leq m\left(r\left(\phi_{\mathcal{F}}^{(\lceil \frac{d}{2} \rceil)}(k\ell), d\right)\right) < \infty$.
- The function $\phi_{\mathcal{F}}^{(\lceil \frac{d}{2} \rceil)}$ satisfies Condition (9), ensuring that $\mathrm{h}(\mathcal{F}) \leq r(b_0,d) = k$.

Since $k \geq \mathrm{h}(\mathcal{F})$, we get $\mathrm{fh}(\mathcal{F}) = \mathrm{fh}_k(\mathcal{F}) < \infty$.

We conclude with a direct consequence of Goaoc et al.'s theorem [1, Theorem 1.2]: if a family $\mathcal{G}$ with ground set $\mathbb{R}^d$ with a finite homological shatter function has a finite fractional Helly number, then this number is at most $d+1$. ◀

Note that even if the topology of the ground set (in our case, $\mathbb{R}^d$) sets the parameter d, the parameter b_0 may vary. For a given $\phi_{\mathcal{F}}^{(\lceil \frac{d}{2} \rceil)}$, we only need to find one value of b_0 such that $\phi_{\mathcal{F}}^{(\lceil \frac{d}{2} \rceil)} \leq \Psi_{d,b_0}$ to apply the theorem. This condition becomes less restrictive as b_0 increases.

5 Example of families with a given homological shatter function

The homological shatter function of a family is always non-decreasing. It turns out that all non-decreasing function can be realized as the homological shatter function of some family.

▶ **Lemma 5.1.** *For any non-decreasing function $f : \mathbb{N} \to \mathbb{N}$, and for every $h \geq 0$ and $d \geq h+2$, there exists a family of sets $\mathcal{F}$ in $\mathbb{R}^d$ such that $\phi_{\mathcal{F}}^{(h)} = f$.*

Proof. Consider countably many disjoint filled boxes in $\mathbb{R}^d$. For $i \geq d-h$, define a family $\mathcal{F}_i := \{F_1^{(i)}, \dots, F_i^{(i)}\}$ as follows. (For $i < d-h$, we can adapt the construction.) Place $f(i)$ disjoint $(d-h-1)$-dimensional polytopes with i facets inside the i-th box, and label the facets of each polytope from 1 to i.

Define the set $F_k^{(i)}$ as the complement of all facets labeled k within the box. The intersection of all members of $\mathcal{F}_i$ is the complement of the boundaries of these polytopes inside the box. By Alexander's duality, the h-th Betti number of this intersection is

$$\tilde{\beta}_h\left(\bigcap_{F\in\mathcal{F}_i} F, \mathbb{Z}_2\right) = f(i).$$

Moreover, for any strict subfamily $\mathcal{F}_i' \subsetneq \mathcal{F}_i$, we have $\tilde{\beta}_{h'}\left(\bigcap_{F\in\mathcal{F}_i'} F, \mathbb{Z}_2\right) = 0$ for all $h' \leq h$. In fact, the only subfamilies of $\mathcal{F} \coloneqq \bigcup_{i\in\mathbb{N}} \mathcal{F}_i$ whose intersection has a nonzero h'-th Betti number for some $h' \leq h$ are precisely the families $\mathcal{F}_i$. It follows that

$$\phi_{\mathcal{F}}^{(h)}(t) = \sup\left\{\tilde{\beta}_h\left(\bigcap_{F\in\mathcal{F}_i} F, \mathbb{Z}_2\right) \mid \#\mathcal{F}_i \leq t\right\} = f(t),$$

since f is non-decreasing. ◀

For every $d > 3$, the integer $h = \lceil \frac{d}{2} \rceil$ satisfies $d \geq h + 2$; the family obtained from Lemma 5.1 satisfies the conditions of Theorem 1.2 despite not satisfying the conditions of [1, Corollary 1.3]. Naturally, we do not need to apply Theorem 1.2 to bound the fractional Helly number of this family; its nerve complex is a disjoint union of simplices, thus the fractional Helly theorem for convex sets in $\mathbb{R}$ can be applied.

Acknowledgments. The author would like to thank Xavier Goaoc for his guidance during the research process, Niloufar Fuladi for her valuable feedback on the writing and the reviewers for their helpful comments.

References

1 Xavier Goaoc, Andreas F. Holmsen, and Zuzana Patáková. Intersection patterns in spaces with a forbidden homological minor, 2024. arXiv:2103.09286.

2 Xavier Goaoc, Pavel Paták, Zuzana Patáková, Martin Tancer, and Uli Wagner. Bounding Helly numbers via Betti numbers. In *A Journey Through Discrete Mathematics: A Tribute to Jiří Matoušek*, pages 407–447. Springer International Publishing, Cham, 2017. doi:10.1007/978-3-319-44479-6_17.

3 Eduard Helly. Über Systeme von abgeschlossenen Mengen mit gemeinschaftlichen Punkten. *Monatshefte für Mathematik und Physik*, 37:281–302, 1930.

4 Andreas F. Holmsen. Large cliques in hypergraphs with forbidden substructures. *Combinatorica*, 40(4):527–537, Aug 2020. doi:10.1007/s00493-019-4169-y.

5 Andreas F. Holmsen and Donggyu Lee. Radon numbers and the fractional Helly theorem. *Israel Journal of Mathematics*, 241(1):433–447, Mar 2021. doi:10.1007/s11856-021-2102-8.

6 Gil Kalai. Combinatorial expectations from commutative algebra. In I. Peeva anv V. Welker, editor, *Combinatorial Commutative Algebra*, volume 1(3), pages 1729–1734. Oberwolfach Reports, 2004. URL: https://api.semanticscholar.org/CorpusID:85558093.

7 Meir Katchalski and Andy Liu. A problem of geometry in $\mathbb{R}^n$. *Proceedings of the American Mathematical Society*, 75(2):284–288, 1979.

8 Friedrich W. Levi. On Helly's theorem and the axioms of convexity. *Journal of the Indian Mathematical Society*, 15:65–76, 1951. URL: https://api.semanticscholar.org/CorpusID:125497683.

9 Jirí Matoušek. Bounded VC-dimension implies a fractional Helly theorem. *Discrete & Computational Geometry*, 31:251–255, 2004.

10 John Milnor. On the Betti numbers of real varieties. *Proceedings of the American Mathematical Society*, 15(2):275–280, 1964.

11 Zuzana Patáková. Bounding Radon number via Betti numbers. *International Mathematics Research Notices*, 04 2024. `doi:10.1093/imrn/rnae056`.

Visualization of Event Graphs for Train Schedules

Johann Hartleb[1], Marie Schmidt[2], Samuel Wolf[2], and Alexander Wolff[2]

1 **DB InfraGO AG, Germany**
`johann.hartleb@deutschebahn.com`

2 **Universität Würzburg, Germany**
`firstname.lastname@uni-wuerzburg.de`

Abstract

Software that is used to compute or adjust train schedules is based on so-called *event graphs*. The vertices of such a graph correspond to *events*; each event is associated with a point in time, a location, and a train. A *train line* corresponds to a sequence of events (ordered by time) that are associated with the same train. The event graph has a directed edge from an earlier to a later event if they are consecutive along a train line. Events that occur at the same location do not occur at the same time.

In this paper, we present a way to visualize such graphs, namely *time-space diagrams*. A time-space diagram is a straight-line drawing of the event graph with the additional constraint that all vertices that belong to the same location lie on the same horizontal line and that the x-coordinate of each vertex is given by its point in time. Hence, it remains to determine the y-coordinates of the locations. A good drawing of a time-space diagram supports users (or software developers) when creating (software for computing) train schedules.

To enhance readability, we aim to minimize the number of turns in time-space diagrams. To this end, we establish a connection between this problem and MAXIMUM BETWEENNESS. Then we develop exact reduction rules to reduce the instance size. We also propose a parameterized algorithm and devise a heuristic that we evaluate experimentally on a real-world dataset.

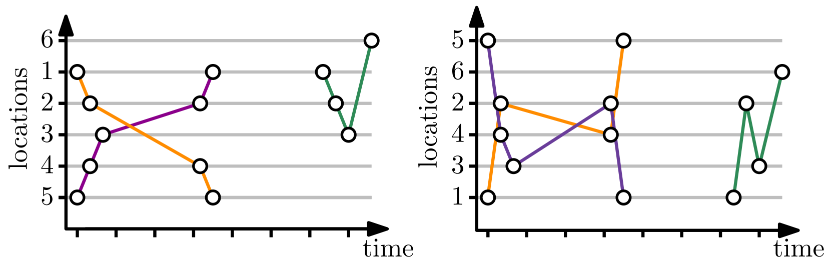
Related Version *Full version:* `https://arxiv.org/abs/2503.01808`

1 Introduction

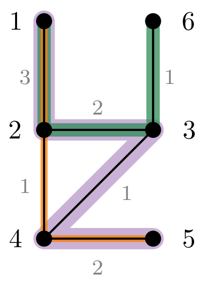
Train schedules are subject to constant changes due to interferences such as temporary infrastructure malfunctions or congestions due to high traffic volume. As a consequence, train schedules must be adjusted in real-time to remedy the disturbances via rerouting and other means. In recent years, the automation of this process has gained track. DB InfraGO AG, a subsidiary of Deutsche Bahn AG, is developing approaches based on a so-called *event graph* [5] as an underlying structure that encodes the necessary information to (re-)compute a train schedule. An event graph models trains running on specific routes on an infrastructure via events. For the further automation and for real-time human intervention, it is important that the event graph can be easily read by humans. For this purpose, we propose a drawing style and algorithms that aim to produce comprehensible drawings of the event graph.

▶ **Definition 1** (Event Graph)**.** An *event graph* $\mathcal{E}$ is a directed graph. Let $V(\mathcal{E})$ denote the vertex set of $\mathcal{E}$. Each vertex v of $\mathcal{E}$, called *event*, is associated with a location $\ell(v)$, a positive integer $\text{train}(v)$, and a point of time $t(v)$ when the event is scheduled. For two different events u and w, if $t(u) = t(w)$, then $\text{train}(u) \neq \text{train}(w)$ and $\ell(u) \neq \ell(w)$. There is an arc (u, w) in $\mathcal{E}$ if (i) $\text{train}(u) = \text{train}(w)$, (ii) $t(u) < t(w)$, and (iii) there is no event v with $\text{train}(v) = \text{train}(u)$ and $t(u) < t(v) < t(w)$.

For a train z, we call the sequence $v_1, \ldots, v_j$ of all events with $\text{train}(v_1) = \cdots = \text{train}(v_j) = z$ ordered by $t(\cdot)$ the *train line* of train z. We propose to visualize event graphs as follows.



(a)



(b)

Figure 1 (a) Two different time-space diagrams of the same event graph $\mathcal{E}$ with locations $\{1, \ldots, 6\}$. (b) The location graph of $\mathcal{E}$; the colored paths are the train lines, the gray numbers the weights.

▶ **Definition 2** (Time-Space Diagram)**.** Let $\mathcal{E}$ be an event graph, let $Y = |\ell(V(\mathcal{E}))|$, and let $y \colon \ell(V(\mathcal{E})) \to \{1, 2, \ldots, Y\}$ be a bijection. The *time-space diagram* induced by y is the straight-line drawing of $\mathcal{E}$ in the plane where event v is mapped to the point $(t(v), y(\ell(v)))$.

In a time-space diagram (see Figure 1a for two examples), we call $y(p)$ the *level* of location p. Given a drawing Γ of an event graph $\mathcal{E}$ and three consecutive events of a train line in $\mathcal{E}$ with pairwise distinct locations p, q, r, we say that there is a *turn* in Γ if the level of q is smaller/larger than the levels of p and r.[1] Experiments suggest that minimizing a classical graph drawing objective, the number of crossings, often does not yield comprehensible drawings. Instead, minimizing the number of turns in a drawing seems to be a promising objective, as Figure 2 illustrates. Therefore, we consider the following problem.

▶ **Problem 3.** *Let $\mathcal{E}$ be an event graph. Find a time-space diagram Γ of $\mathcal{E}$ that minimizes the number of turns along the train lines in Γ.*

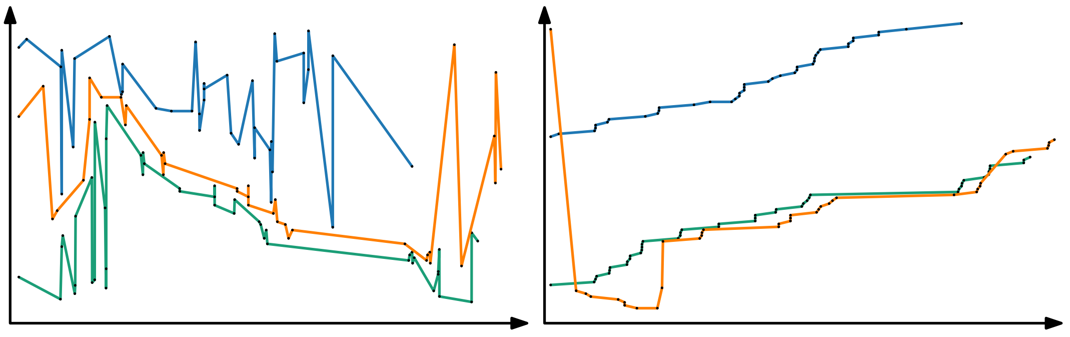


Figure 2 Two time-space diagrams of the same event graph. Left: A crossing-minimal drawing with zero crossings (and 71 turns). Right: A turn-minimal drawing with one turn (and five crossings).

Note that the numbers of turns in a time-space diagram is determined solely by the function y, which represents an ordering of the locations. Therefore, Problem 3 is closely related to the following problem.

[1] Note that this definition does not consider the case where consecutive events have the same location. It is easy to see, however, that we can normalize the graph such that consecutive events always have different locations without changing the optimal solution of Problem 3.

▶ **Problem 4** (Maximum Betweenness). *Let S be a finite set, and let $R \subseteq S \times S \times S$ be a finite set of ordered triplets called* constraints. *A total order $\prec$* satisfies *a constraint $(a,b,c) \in R$ if either $a \prec b \prec c$ or $c \prec b \prec a$ holds. Find a total order that maximizes the number of satisfied constraints.*

Note that there is a one-to-one correspondance between (optimal) solutions of Problems 3 and 4. Maximum Betweenness is NP-hard [10], which implies the NP-hardness of Problem 3. Maximum Betweenness has been studied extensively [4, 11–13]. In particular, it admits 1/2-approximation algorithms [2, 9], but for any $\varepsilon > 0$ it is NP-hard to compute a $(1/2+\varepsilon)$-approximation [1]. Note that, due to the different objectives, these (non-) approximability results do not carry over to Problem 3; see [8] for details.

When translating to Maximum Betweenness, we lose information about the train lines. However, this information proves to be beneficial for our purposes. In particular, we want to leverage the natural sparseness of train infrastructures. We define two auxiliary graphs that capture the connections between locations in $\mathcal{E}$, which we use in our algorithms.

▶ **Definition 5** (Location Graph). Let $\mathcal{E}$ be an event graph. The *location graph* $\mathcal{L}$ of $\mathcal{E}$ is an undirected weighted graph whose vertices are the locations of $\mathcal{E}$. For two locations $p \neq q$, the weight $w(\{p,q\})$ of the edge $\{p,q\}$ in $\mathcal{L}$ corresponds to the number of arcs (u,v) or (v,u) in the event graph $\mathcal{E}$ such that $\ell(u) = p$ and $\ell(v) = q$ and $\text{train}(u) = \text{train}(v)$. If $w(\{p,q\}) = 0$, then p and q are not adjacent in $\mathcal{L}$.

See Figure 1b for an example. Note that the train line of a train z in the event graph corresponds to a walk (a not necessarily simple path) in the location graph. Abusing the notation of a train line, we also call this path in the location graph a *train line* of z.

▶ **Definition 6** (Augmented Location Graph). Let $\mathcal{E}$ be an event graph. The *augmented location graph* $\mathcal{L}'$ of $\mathcal{E}$ is a supergraph of the location graph $\mathcal{L}$ of $\mathcal{E}$ containing additional edges $\{\ell(v_{i-1}), \ell(v_{i+1})\}$ for each consecutive triplet (v_{i-1}, v_i, v_{i+1}) of a train line in $\mathcal{E}$.

The augmented graph $\mathcal{L}'$ has the crucial property that three consecutive events v_{i-1}, v_i, v_{i+1} of a train line whose locations can cause a potential turn form a triangle in $\mathcal{L}'$.

2 Exact Algorithms

An exact reduction rule. There are substructures that can be solved easily and independently. Consider the location graph $\mathcal{L}$ of an event graph $\mathcal{E}$. We call a vertex $p \in V(\mathcal{L})$ a *terminal* if a train starts or ends at p, and we say that a path in $\mathcal{L}$ is a *chain* if each of its vertices has degree exactly 2 in $\mathcal{L}$ and the path cannot be extended without violating this property. If a chain contains no terminals, and trains move through the entire chain without turning back, then there is always a turn-minimal drawing of $\mathcal{E}$ that contains no turn along the chain. This is due to the fact that any turn on the chain can be moved to a non-chain vertex adjacent to one of the chain endpoints.

We now sketch a generalization of this intuition. We call two vertices whose removal disconnects a graph a *separating pair*. Let $\{s,t\}$ be a separating pair of $\mathcal{L}$, let C be a connected component of $\mathcal{L} - \{s,t\}$, and let $C' = \mathcal{L}[V(C) \cup \{s,t\}]$. We call C a *transit component* if C does not contain any terminal, and if the trains passing through C' via s also pass through t (before possibly passing through s again).

A transit component C is *contractible* if, for each path associated to a train line in C, we can assign a direction such that the resulting directed graph is acyclic; see in Figure 3.

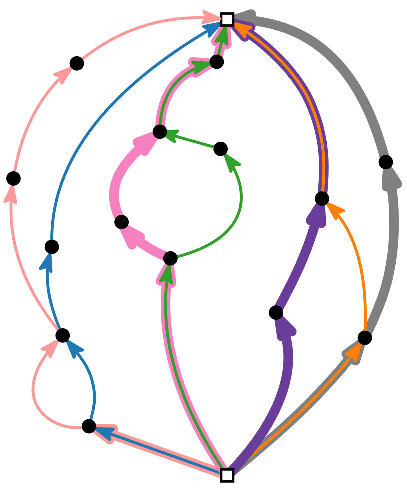


Figure 3 A transit component that is part of a larger location graph. Each train is represented by a sequence of arcs of the same color. Some trains share edges. The component is contractible with respect to the set of trains.

▶ **Reduction Rule 1** (Transit Component Contraction). Let $\mathcal{E}$ be an event graph, let $\mathcal{L}$ be its location graph, and let C be a contractible transit component that is separated by $\{s, t\}$. For each train z that traverses C, replace in $\mathcal{E}$ the part of the train line of z between the events that correspond to s and t by the arc (directed according to time) that connects the two events.

▶ **Theorem 7.** *Let $\mathcal{E}$ be an event graph. If $\mathcal{E}'$ is an event graph that results from applying Reduction Rule 1 to $\mathcal{E}$, then a turn-optimal drawing of $\mathcal{E}$ and a turn-optimal of $\mathcal{E}'$ have the same number of turns.*

Proof sketch. Let Γ be a turn-optimal drawing of $\mathcal{E}$, and let Γ' be a turn-optimal drawing of $\mathcal{E}$ after applying Reduction Rule 1 once; to a contractible transit component C. We now turn Γ' into a drawing of $\mathcal{E}$ without introducing new turns. We expand the mapping y' corresponding to Γ' by placing C between the levels of $y'(s)$ and $y'(t)$ such that the ordering of the locations in C respects a topological ordering. Due to the topological ordering, C contains no turns, whereas turns at s and t are preserved.

Conversely, we can transform Γ into a drawing of $\mathcal{E}'$ without changing the number of turns. Since C contains no terminal, we can move any turn in C to one of the two locations s and t that separate C from the rest of the location graph, and then replace every train line that traverses C by a straight-line segment connecting the two events that correspond to s and t (on that train line). For a complete proof, see [8]. ◀

A parameterized algorithm. We use dynamic programming on a nice tree decomposition of the augmented location graph $\mathcal{L}'$. For a definition of a (nice) tree decomposition, see [3]. Since three consecutive locations of a train line form a triangle in $\mathcal{L}'$, there must be a bag that contains all three locations. Hence, it suffices to check each bag for potential turns in order to find a drawing with the minimum number of turns. For the full proof, see [8].

▶ **Theorem 8.** *Let $\mathcal{E}$ be an event graph, and let $\mathcal{L}'$ be its augmented location graph. Computing a turn-optimal time-space diagram of $\mathcal{E}$ is fixed-parameter tractable with respect to the treewidth of $\mathcal{L}'$.*

3 A Greedy Heuristic Approach

The heuristic tries to minimize the number of turns by iteratively inserting trains into an ordering of locations such that the inserted train does not violate the existing ordering and such that its train line is as monotone as possible. The idea is as follows.

Given an event graph $\mathcal{E}$ and its associated location graph $\mathcal{L}$ with train lines $Z = \{z_1, \dots, z_k\}$, we sort Z with respect to the total weight of the edges of the train lines in $\mathcal{L}$, in descending order.

For simplicity, we assume that each train line is a simple path in $\mathcal{L}$. If this is not the case, we decompose each non-simple train line into multiple simple train lines. We construct a directed auxiliary graph G that is initially empty and to which we iteratively insert the edges of each train line until we obtain a directed version of $\mathcal{L}$. For a directed graph (or simply a set of arcs) H, let $E(H)$ be the set of undirected edges that correspond to the arcs of H.

Let $i \in \{1, \dots, k\}$ the index of the current iteration. For each connected component P of $E(z_i) \setminus E(G)$, we do the following. Note that P is adjacent to at most two vertices of G. If P is adjacent to *exactly* two vertices p and q in G, we test whether there is a q–p path in G. If this is the case, we select among the edges of P one that has the smallest weight in $\mathcal{L}$. We reverse the selected edge. Then we insert the vertices and the directed edges of P into G. (The reversal ensures that G remains acyclic.)

After the last iteration, we compute a topological order π of G and draw the time-space diagram of $\mathcal{E}$ induced by π. See [8] for the pseudocode of the heuristic.

4 Experimental Analysis

We test the effectiveness of the reduction rule and of the heuristic on an anonymized and perturbed dataset with 19 instances provided by DB InfraGO AG; see Figure 4 for an overview of the dataset. The result of each experiment is the average value over 25 repetitions of the experiment. We implemented our algorithms in the programming language `Python`. We used Networkx [7] to handle most of the graph operations and Gurobi [6] to solve the integer linear program. All experiments were conducted on a workstation running Fedora 40 with Kernel 6.10.6 using an Intel-7-8850U CPU with 16GB RAM.

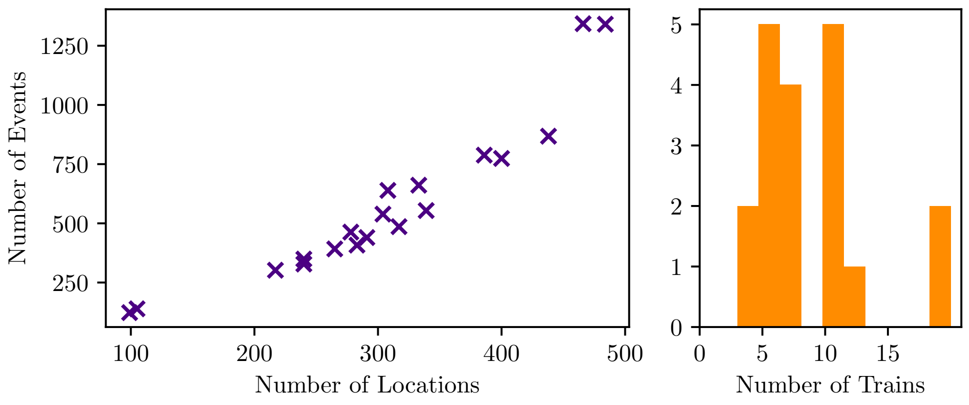


Figure 4 Left: Instances with respect to the number of events and the number of locations. Right: The histogram depicts the frequency (y-axis) of the number of trains (x-axis) in the dataset, i.e., there are 5 instances containing 5 trains.

Effectiveness of the reduction rule. Our implementation of Reduction Rule 1 is restricted to exhaustively contract chains. But even with this restriction, the reduction rule proves to be effective on the provided dataset. On average, the number of locations was reduced by 75%, where the best result was a reduction by 89% (instance 50_4) and the worst result was a reduction by 55% (instance 20_3). A full evaluation is shown in Figure 5.

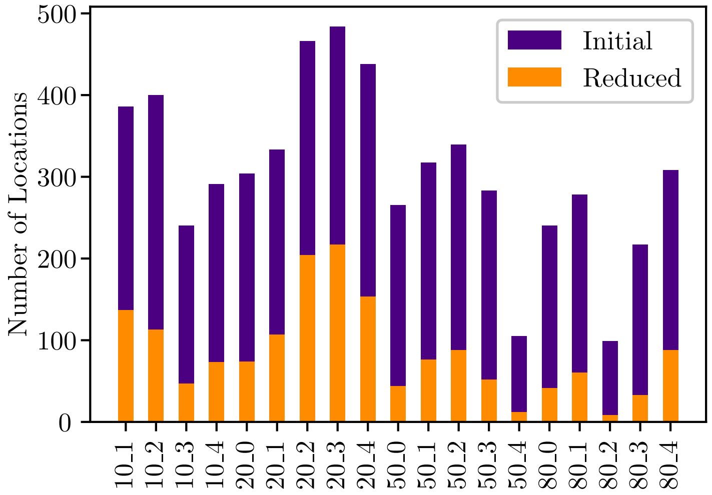


Figure 5 Results of the effectiveness of applying contractions, restricted to contracting chains. The bar diagram shows the number of locations in each instance before and after contraction.

Effectiveness of the heuristic. We compare the results of our heuristic to the corresponding optima, which we obtained using the integer linear program (ILP) described in [8], and to the heuristic EM-algorithm that Filipović, Kartelj, and Matić [4] developed for the Maximum Betweenness problem. We use the implementation of the EM-algorithm of Filipović et al. [4] and their parameter settings. On average, our algorithm yields results that are four times better than those of the EM-algorithm on the original instances. On the reduced instances, our algorithm loses its advantage; there, the two approaches perform comparably on average. However, our approach is significantly faster, as our algorithm takes 0.2 s on the slowest instance while the EM-algorithm takes 26.3 s. See Figures 6 and 7 for a detailed overview of the effectiveness and runtime, respectively. Note that we show the runtime of the ILP approach only on the reduced instances since the ILP did not converge on most of the original instances within a time limit of one hour.

5 Conclusion and Future Work

In this work we have introduced the problem of computing turn-minimal time-space diagrams to visualize event graphs. We have established a connection between this problem and Maximum Betweenness, we have presented an FPT-algorithm parameterized by the treewidth of the location graph, and we have proposed a heuristic for solving large instances.

The heuristic produces drawings that are still far from optimal, thus it would be interesting to improve this heuristic or to find a better one. As part of our ongoing research, we have developed a new exact integer linear programming approach that shows promising preliminary results, solving the largest real-world instance in less than a second.

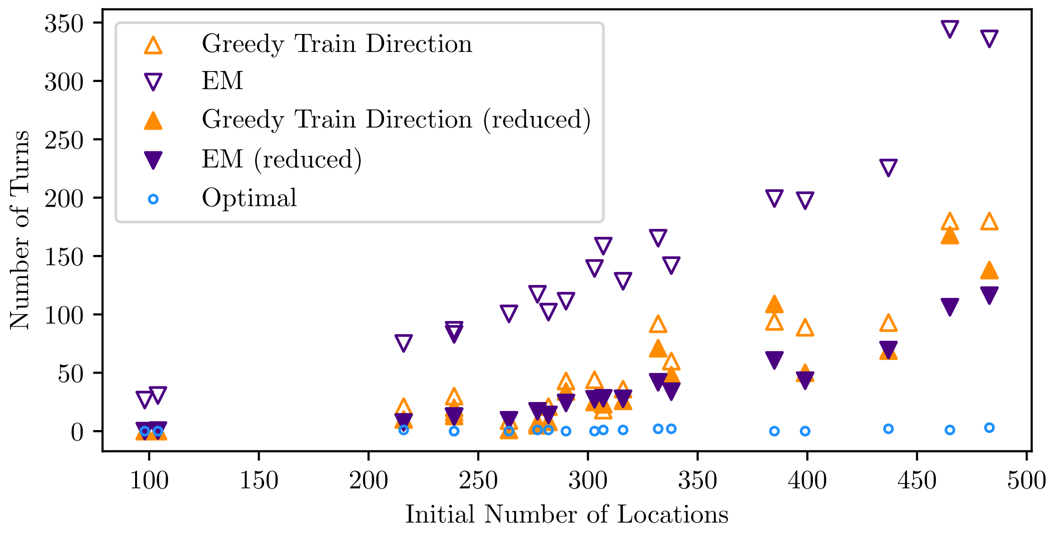


Figure 6 Comparison of the effectiveness between our heuristic (Greedy Train Direction), the EM-algorithm of Filipović et al. [4], and the optimum; on the original and the reduced instances.

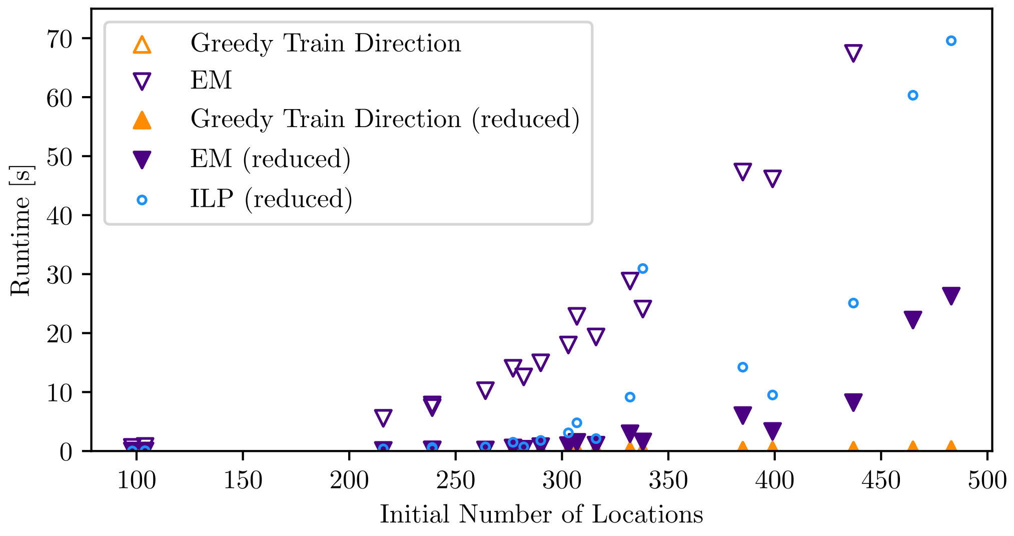


Figure 7 Runtime comparison between our heuristic (Greedy Train Direction) and the EM-algorithm of Filipović et al. [4] on the original and the reduced instances.

References

1 Per Austrin, Rajsekar Manokaran, and Cenny Wenner. On the NP-hardness of approximating ordering-constraint satisfaction problems. *Theory of Computing*, 11(10):257–283, 2015. doi:10.4086/toc.2015.v011a010.

2 Benny Chor and Madhu Sudan. A geometric approach to betweenness. *SIAM Journal on Discrete Mathematics*, 11(4):511–523, 1998. doi:10.1137/S0895480195296221.

3 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 1st edition, 2015. doi:10.5555/2815661.

4 Vladimir Filipović, Aleksandar Kartelj, and Dragan Matić. An electromagnetism metaheuristic for solving the maximum betweenness problem. *Applied Soft Computing*, 13(2):1303–1313, 2013. doi:10.1016/j.asoc.2012.10.015.

5 Rihab Gorsane, Khalil Gorsan Mestiri, Daniel Tapia Martinez, Vincent Coyette, Beyrem Makhlouf, Gereon Vienken, Minh Tri Truong, Andreas Söhlke, Johann Hartleb, Amine Kerkeni, Irene Sturm, and Michael Küpper. Reinforcement learning based train rescheduling on event graphs. In *26th Int. Conf. Intell. Transport. Syst. (ITSC)*, pages 874–879. IEEE, 2023. doi:10.1109/ITSC57777.2023.10422531.

6 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL: https://www.gurobi.com.

7 Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using NetworkX. In *7th Python Sci. Conf.*, pages 11–15. SciPy, 2008. doi:10.25080/TCWV9851.

8 Johann Hartleb, Marie Schmidt, Samuel Wolf, and Alexander Wolff. Visualization of event graphs for train schedules. arXiv report, 2025. URL: https://arxiv.org/abs/2503.01808.

9 Yury Makarychev. Simple linear time approximation algorithm for betweenness. *Operations Research Letters*, 40(6):450–452, 2012. doi:10.1016/j.orl.2012.08.008.

10 Jaroslav Opatrny. Total ordering problem. *SIAM Journal on Computing*, 8(1):111–114, 1979. doi:10.1137/0208008.

11 Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991. doi:10.1016/0022-0000(91)90023-X.

12 Aleksandar Savić. On solving the maximum betweenness problem using genetic algorithms. *Serdica Journal of Computing*, 3(3):299–308, 2009. doi:10.55630/sjc.2009.3.299-308.

13 Aleksandar Savić, Jozef Kratica, Marija Milanović, and Djordje Dugošija. A mixed integer linear programming formulation of the maximum betweenness problem. *European Journal of Operational Research*, 206(3):522–527, 2010. doi:10.1016/j.ejor.2010.02.028.

Dynamic Balanced Clique Separators for Disk Graphs

Alexander Baumann[1], Katharina Klost[1], Kristin Knorr[1], and Wolfgang Mulzer[1]

1 **Institut für Informatik, Freie Universität Berlin, Berlin, Germany**
`[akauer,kathklost,knorrkri,mulzer]@inf.fu-berlin.de`

Abstract

Let S be a set of n *sites* in the plane, where each site $s \in S$ has an *associated radius* $r_s > 0$, and let $\mathcal{D}(S)$ be the *disk intersection graph* defined on S. A subset $S_{\text{sep}} \subseteq S$ is a *balanced separator* if $S \setminus S_{\text{sep}}$ can be separated into two subsets S_1, S_2 the size of each being at most a constant fraction of n with no edges between them.

We describe a data structure for maintaining a balanced separator consisting of $O(\sqrt{n})$ *cliques* under insertions and deletion of sites in S, where each update requires $O(\log^2 n)$ amortized time. The data structure requires $O(n \log n)$ space.

1 Introduction

Balanced separators divide graphs into three parts: two parts whose size is at most a constant fraction of the number of vertices in the graph which only have outgoing edges towards the third part, the separator. As such, they directly lead to divide-and-conquer approaches and thus play an integral part in many graph-based algorithms and data structures [3, 7, 9, 10, 12].

For most geometric intersection graphs balanced separators can be arbitrary large, as these graphs can contain arbitrary large cliques. Thus, using separators in this context requires bounding other aspects apart from only the size to efficiently use balanced separators. For example, for disk intersection graphs there exist separators consisting of $O(\sqrt{n})$ cliques [1]. This was recently used by Chan and Huang [5] (in their full version [6]) in a subroutine to dynamically maintain a disk intersection graph under insertions and deletions. In their work they use a rebuilding approach—they repeatedly build static separators.

In this work we extend a result for computing a static clique-based balanced separator for intersection graphs of certain fat geometric objects by de Berg et al. [1] into a dynamic data structure for disk intersection graphs. Our data structure maintains a balanced separator consisting of $O(\sqrt{n})$ cliques under insertions and deletions of sites. These updates require $O(\log^2 n)$ amortized time each and the data structure requires $O(n \log n)$ space. While this data structure follows a rebuilding approach as well, it allows updates in between the rebuilds.

2 Preliminaries

Let S be a set of point sites in the plane with associated radii r_s. We will consider the sites as disks with center s and radius r_s and often do not distinguish between a site and the associated disk. The *disk intersection graph* $\mathcal{D}(S)$ of S has vertex set S and two sites are adjacent if and only if their disks intersect.

▶ **Definition 2.1.** A subset $S_{\text{sep}} \subseteq S$ is a *β-balanced separator* for $\mathcal{D}(S)$, if $S \setminus S_{\text{sep}}$ can be partitioned into two subsets S_1, S_2 with no edges between them and $\max\{|S_1|, |S_2|\} \le \beta n$.

For a decomposition $\mathcal{C}(S_{\text{sep}})$ of S_{sep} into cliques, and a given weight function γ on the cliques, we define the *weight* of the decomposition as $w(S_{\text{sep}}) := \sum_{C \in \mathcal{C}(S_{\text{sep}})} \gamma(|C|)$.

We assume the real RAM [11] as computational model. This requires some careful considerations [4] when using quadtrees in Section 4, as no floor function is available.

3 Balanced Clique Separators by de Berg et al.

In this section, we summarize the separators defined by de Berg et al. [1]. They construct balanced clique separators of small weight in the more general setting of intersection graphs of certain fat objects in arbitrary dimension. We will however focus on the easier case of disks in the plane and do some slight alterations in preparation for the later extension.

Let H_0 be a square of minimum side length, such that H_0 completely contains at least $\frac{n}{50}$ sites from S.[1] Without loss of generality, we assume that H_0 is a square with side length 1, centered at the origin. Subdivide the plane with a grid $\mathcal{G}_0$ centered at the origin, where each grid cell has side length $\frac{1}{\sqrt{n}}$. Then, H_0 is the $\sqrt{n} \times \sqrt{n}$ neighborhood of the origin. Let H_i be the $(\sqrt{n}+2i) \times (\sqrt{n}+2i)$ neighborhood of the origin for $i = 1, \dots, \sqrt{n}$. Each H_i has side length $1 + \frac{2i}{\sqrt{n}}$. Note that $H_{\sqrt{n}}$ has a side length of 3.

Each H_i defines a partition of S into the sites $S_{\text{in}}(H_i)$ whose disks are completely contained in H_i, the sites $S_{\text{out}}(H_i)$ whose disks are completely outside H_i, and the sites $S_\partial(H_i)$ whose disks intersect the boundary of H_i. Hence, the set $S_\partial(H_i)$ is a separator for each $i = 0, \dots, \sqrt{n}$.

A disk is called *large*, if its radius is at least $\frac{1}{4}$ and small otherwise. Let S_{large} be the set of sites with a large associated disk that intersect $H_{\sqrt{n}}$. We define the set of *candidate separators* as the set of all $S_{\text{sep}}(H_i) := S_\partial(H_i) \cup S_{\text{large}}$.

▶ **Lemma 3.1** (Lemma 2.1 in [1]). *For any $0 \le i \le \sqrt{n}$ we have:*

$$\max\left(|S_{\text{in}}(H_i)|, |S_{\text{out}}(H_i)|\right) < \frac{49}{50}n$$

Proof. First, consider $|S_{\text{out}}(H_i)|$. There are at least $\frac{n}{50}$ sites in H_0. These sites are either in $S_{\text{in}}(H_i)$ or in $S_\partial(H_i)$ for all $i = 0, \dots, \sqrt{n}$. Thus, each $S_{\text{out}}(H_i)$ contains at most $\frac{49}{50}n$ sites. Secondly, we bound $|S_{\text{in}}(H_i)|$. Consider a grid $\mathcal{G}_\times$ with arbitrary center and cell size $\frac{1}{2}$. Since H_0 is the smallest square containing at least $\frac{n}{50}$ sites, each of the at most 7×7 cells of $\mathcal{G}_\times$ covering H_i contains less than $\frac{n}{50}$ sites and the claim follows. ◀

Now that we know that we have a set of $O(\sqrt{n})$ balanced candidate separators, we define a clique decomposition for each of them in such a way, that at least one of the clique decompositions will have small weight.

Consider all sites with a small associated disk that can be part of a separator, namely $S^* := S \setminus (S_{\text{in}}(H_0) \cup S_{\text{out}}(H_{\sqrt{n}}) \cup S_{\text{large}})$. We partition S^* into *size classes* based on the radii of the disks. For integers r with $1 \le r \le r_{\max}$ where $r_{\max} := \lceil \log n/2 \rceil - 2$, define:

$$S_r^* := \left\{ s \in S^* \;\middle|\; \frac{2^{r-1}}{\sqrt{n}} \le r_s < \frac{2^r}{\sqrt{n}} \right\}$$

Furthermore, let S_0^* be the set of all disks in S^* with a radius smaller than $\frac{1}{\sqrt{n}}$.

[1] In the original proof of de Berg et al. [1] the minimum square contained the complete objects. Using only the sites is a variant of Chan and Huang [5] introduced in their full version [6].

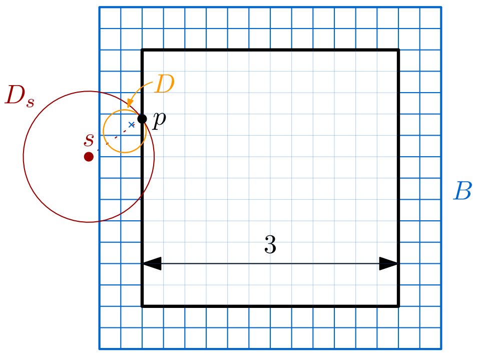


Figure 1 The situation when s of a large disk D_s is not contained in B. In this case, the disk D with radius $\frac{1}{4}$ contained in D_s is completely contained in B and contains the center of a cell.

▶ **Lemma 3.2.** *All sets S_r^* with $1 \le r \le r_{\max}$ can be decomposed into at most $O\left(\frac{n}{2^{2r}}\right)$ cliques.*

Proof. For a fixed r, consider a grid $\mathcal{G}_{r-1}$ with cells of side length $\frac{2^{r-1}}{\sqrt{n}}$ centered at the origin. Note that $\mathcal{G}_0$ is a subdivision of $\mathcal{G}_{r-1}$. Then, $H_{\sqrt{n}}$ is covered by the union of the cells of the $\frac{3\sqrt{n}}{2^{r-1}} \times \frac{3\sqrt{n}}{2^{r-1}}$ subgrid of $\mathcal{G}_{r-1}$ centered at the origin. As some sites in S_r^* can lie outside $H_{\sqrt{n}}$, we also have to consider them. As the radius of their disks is less than $\frac{2^r}{\sqrt{n}}$, it suffices to extend the subgrid to a size of $(\frac{3\sqrt{n}}{2^{r-1}} + 4) \times (\frac{3\sqrt{n}}{2^{r-1}} + 4)$ to contain all sites of S_r^*.

All disks of sites of S_r^* which lie in the same cell of $\mathcal{G}_{r-1}$ are stabbed by the center of the cell and thus form a clique. ◀

▶ **Lemma 3.3.** *S_{large} can be decomposed into $O(1)$ cliques $\mathcal{C}(S_{\text{large}})$.*

Proof. Let B be a square with side length 4 centered at the origin. Furthermore, consider a subdivision of B into $O(1)$ cells Σ of side length $\frac{1}{4}$. If $s \in S_{\text{large}}$ is contained in B, then, as above, the center of the cell that contains s stabs its disk D_s. In the other case, the radius of D_s is larger than $\frac{1}{2}$ as $D_s \cap H_{\sqrt{n}} \neq \emptyset$. Let p be a point on the boundaries of both D_s and $H_{\sqrt{n}}$ and let sp be the line segment between s and p. Then the disk D with radius $\frac{1}{4}$ with its center on sp that has p on its boundary is completely contained in $D_s \cap B$. Furthermore, D contains the center of the cell $\sigma \in \Sigma$ that contains the center of D. See Figure 1. Hence, all disks associated with S_{large} are stabbed by centers of Σ. ◀

Each site in S_0^* will form its own clique. Let $\mathcal{C}(S^*)$ be the set of all cliques discussed above. $\mathcal{C}(S^*)$ induces a clique decomposition on each of the candidate separators $S_{\text{sep}}(H_i)$. Note that $S_{\text{sep}}(H_i)$ might contain only subcliques of the cliques defined by Lemma 3.2. Let $\hat{S}_{\text{sep}}(H_i)$ be the separator $S_{\text{sep}}(H_i)$ where each partial clique was extended to the respective complete clique of $\mathcal{C}(S^*)$. The following Lemma 3.4 implies that at least one $\hat{S}_{\text{sep}}(H_i)$ is a balanced separator with weight $O(\sqrt{n})$.

▶ **Lemma 3.4** (Lemma 2.3 in [1]). *If $\gamma(t) \in O(t^{1/2-\varepsilon})$, $\varepsilon > 0$, then $\sum_{i=1}^{\sqrt{n}} w(\hat{S}_{\text{sep}}(H_i)) \in O(n)$.*

4 Efficient Computation and Maintenance of the Separators

In this section we will describe how a separator with $\gamma(t) = 1$ can be found with the help of static data structures. We will then use these data structures to maintain a separator under insertions and deletions of sites. See Figure 2 for an overview of the components.

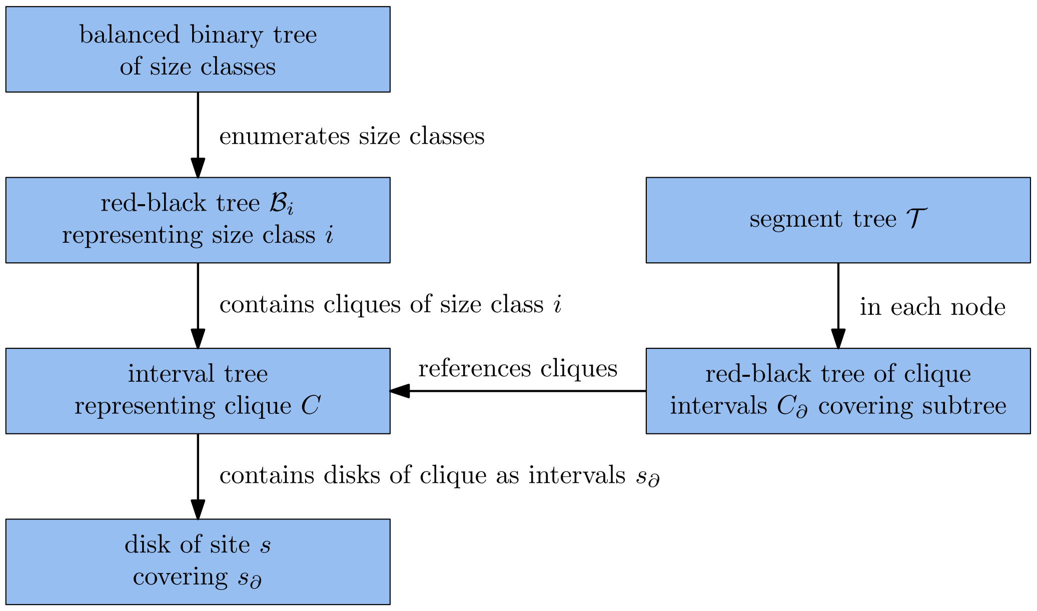

balanced binary tree of size classes
enumerates size classes
red-black tree $\mathcal{B}_i$ representing size class i
segment tree $\mathcal{T}$
contains cliques of size class i
in each node
interval tree representing clique C
references cliques
red-black tree of clique intervals C_∂ covering subtree
contains disks of clique as intervals s_∂
disk of site s covering s_∂

Figure 2 The data structures for obtaining the separator. An arrow between two components indicates references from one to the other.

4.1 Obtaining a Static Separator

First, we need the square H_0 as described above. It can be found by building a compressed quadtree on the sites. We use the algorithm described by Buchin et al. [4] which constructs the quadtree in $O(n \log n)$ time on a real RAM, but has the disadvantage that compressed cells might not be aligned on a global grid. This tree has size $O(n)$. After building the compressed quadtree, we can find a smallest quadtree cell H_0 that contains at least $\frac{n}{50}$ sites in $O(n)$ time. With this H_0 the proof for Lemma 3.1 can be still applied, as each H_i intersects at most 49 quadtree cells with a size in $[\frac{1}{2}, 1)$ (with larger leaf cells or compressed cells split further to reach that range).

Once the square H_0 is known, we can find the square $H_{\sqrt{n}}$ and the set of all disks that intersect $H_{\sqrt{n}}$ in $O(n)$ time. We will only continue with these disks, as other disks cannot be part of a separator. For each size class r with $1 \le r \le r_{\max}$ we maintain a red-black search tree [8] $\mathcal{B}_r$ that stores all cliques in lexicographical order by their associated cells of $\mathcal{G}_{r-1}$. Similarly, we maintain a red-black tree $\mathcal{B}_{\text{large}}$ storing the cliques of S_{large} by their stabbing point and a red-black tree $\mathcal{B}_0$ storing the sites of S_0^* as singleton cliques. Obtaining the cells used to define a clique and constructing these trees requires $O(n \log n)$ overall time. The trees $\mathcal{B}_r$ are stored in a balanced binary search tree for retrieval. In order to find the clique decomposition for each $\hat{S}_{\text{sep}}(H_i)$, we observe:

▶ Observation 4.1. For each $s \in S$, the set $s_\partial = \{i \mid s \in S_\partial(H_i)\}$ is an interval that is a subset of $\{0, \dots, \sqrt{n}\}$. This interval can be found in $O(\log n)$ time for a fixed site.

The next step is to extend this observation to cliques. Let C be a clique. Then we define

$$C_\partial = \bigcup_{s \in C} s_\partial$$

Each clique C is represented by an interval tree [8] based on a red-black tree where each disk is inserted as its interval s_∂. This interval tree allows us to find the interval C_∂ in constant time after construction and can be built in $O(|C| \log |C|)$ time. For a site s it also allows us to find the intervals $C_\partial \cup s_\partial$ and $(C \setminus \{s\})_\partial$ in $O(\log |C|)$ time.

It remains to find the clique separator $\hat{S}_{\text{sep}}(H_i)$ of minimal size. This is equivalent to finding the i which minimizes the number of sets C_∂ with $i \in C_\partial$, excluding the cliques of S_{large}. For this we construct a perfect binary tree $\mathcal{T}$ over the leaves $0, \dots, \sqrt{n}$ which is treated as a segment tree [2]. To store the covering intervals of each node we use red-black trees as well, which hold pointers to the respective cliques. Additionally, we store in each node of $\mathcal{T}$ the minimum number of intervals on the path to any leaf (including the respective node). Then, the intervals C_∂, excluding $\mathcal{B}_{\text{large}}$, are inserted into $\mathcal{T}$. As a property of segment trees, each interval is added to the red-black trees of at most $O(\log n)$ nodes, requiring $O(\log^2 n)$ updates to node minimums. Overall, this requires $O(n \log^2 n)$ time and $O(n \log n)$ space.

This tree can now be used to find a balanced clique separator by traversing the tree from the root to a leaf, always descending into the node with the smallest count associated to it, collecting all sets of cliques encountered along the way. Additionally, all cliques of $\mathcal{B}_{\text{large}}$ must be added. Obtaining the $O(\log n)$ sets of cliques requires $O(\log n)$ time. Following Lemma 3.4, merging all (disjoint) sets of cliques yields $O(\sqrt{n})$ cliques.

▶ **Lemma 4.2.** *Given n sites with associated radius in the plane, then a $\frac{49}{50}$-balanced clique separator for $\mathcal{D}(S)$ can be found in $O(n \log^2 n)$ time using $O(n \log n)$ space.*

4.2 Maintaining a Dynamic Separator

For Lemma 3.1 we require that H_0 is the smallest square (or quadtree cell) containing $\frac{n}{50}$ sites. When adding or removing sites we cannot guarantee this anymore. Hence, to implement updates for the data structure described in Section 4.1, we periodically rebuild the data structure from scratch. In between the rebuilds we update the internal data structures and the separator as well. That way we can maintain a weaker separator.

▶ **Lemma 4.3.** *Let S be a set of n sites and H_0 as above. After adding or removing up to $\frac{n}{99}$ sites, resulting in n' sites total, we have for any $0 \le i \le \sqrt{n}$:*

$$\max\left(|S_{\text{in}}(H_i)|, |S_{\text{out}}(H_i)|\right) < \frac{99}{100} n'.$$

Let n be the size after the last rebuild of the data structure. When inserting a site s, it is first checked if the number of updates since the last rebuild would be above $\frac{n}{99}$. If this is the case, then the complete data structure is rebuild including the new disk according to Lemma 4.2. This requires $O(\log^2 n)$ amortized time per insertion. When no rebuild occurred it is checked whether s intersects $H_{\sqrt{n}}$. If not, then we are done. Otherwise, the tree $\mathcal{B}_r$ for the size class r of s is used to identify the clique C that s belongs to. If the clique does not exist, an empty interval tree is added as C to $\mathcal{B}_r$. Then, s_∂ is inserted into the interval tree. When s is not a large disk, then in addition $\mathcal{T}$ is updated regarding the extended interval C_∂. This involves removing C_∂ from all previous nodes and inserting the extended C_∂ anew, while updating the node minimums accordingly. Afterwards, the separator is updated if required by following the minimum path in $\mathcal{T}$. Updating $\mathcal{T}$ including its nodes requires $O(\log^2 n)$ time and the remaining steps require $O(\log n)$ time.

Deleting a site s follows the same general steps and requires $O(\log^2 n)$ amortized time.

▶ **Theorem 4.4.** *There is a data structure of size $O(n \log n)$ which maintains a $\frac{99}{100}$-balanced separator of $O(\sqrt{n})$ cliques for the disk intersection graph of n sites in the plane with associated radius. Insertions or deletions of sites require $O(\log^2 n)$ amortized time.*

Note that the separator of Theorem 4.4 is maintained implicitly as a set of $O(\log n)$ sets of cliques—actually constructing the separator requires $O(\sqrt{n})$ time. In case the actual separator is required, the data structure of Theorem 4.4 can be used with a hysteresis approach. For this, the data structure is updated as usual, but the separator update is only applied if the number of cliques in the previously chosen separator grows by $O(\sqrt{n})$ or a global rebuild occurs.

Adjusting Theorem 4.4 for a non-constant weight function γ is straightforward. When $\gamma(t)$ can be computed in $O(\log^2 t)$ time, then the bounds of Theorem 4.4 still apply.

Similarly, the approach for convex fat objects of de Berg et al. [1] is also applicable.

References

1 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A Framework for Exponential-Time-Hypothesis–Tight Algorithms and Lower Bounds in Geometric Intersection Graphs. *SIAM J. Comput.*, 49(6):1291–1331, 2020. doi:10.1137/20M1320870.

2 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, 3rd edition, 2008. doi:10.1007/978-3-540-77974-2.

3 Édouard Bonnet, Sergio Cabello, and Wolfgang Mulzer. Maximum matchings in geometric intersection graphs. *Discrete Comput. Geom.*, 70(3):550–579, 2023. doi:10.1007/s00454-023-00564-3.

4 Kevin Buchin, Maarten Löffler, Pat Morin, and Wolfgang Mulzer. Preprocessing imprecise points for Delaunay triangulation: Simplified and extended. *Algorithmica*, 61(3):674–693, 2011. doi:10.1007/s00453-010-9430-0.

5 Timothy M. Chan and Zhengcheng Huang. Dynamic Geometric Connectivity in the Plane with Constant Query Time. In *40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 36:1–36:13, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2024.36.

6 Timothy M. Chan and Zhengcheng Huang. Dynamic geometric connectivity in the plane with constant query time, 2024. doi:10.48550/arXiv.2402.05357.

7 Timothy M. Chan and Dimitrios Skrepetos. Approximate Shortest Paths and Distance Oracles in Weighted Unit-Disk Graphs. In *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:13, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2018.24.

8 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* MIT Press, 3rd edition, 2009.

9 Philip Klein and Shay Mozes. Optimization algorithms for planar graphs. draft, https://planarity.org.

10 Wolfgang Mulzer and Max Willert. Compact Routing in Unit Disk Graphs. In *31st International Symposium on Algorithms and Computation (ISAAC 2020)*, volume 181 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ISAAC.2020.16.

11 Franco P. Preparata and Michael Ian Shamos. *Computational Geometry - An Introduction.* Springer, 1985. doi:10.1007/978-1-4612-1098-6.

12 Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, November 2004. doi:10.1145/1039488.1039493.

A curve with rotation number one that is not universal for beacon routing

Florestan Brunck[1], Maarten Löffler[2], and Rodrigo I. Silveira[3]

1 University of Copenhagen, Denmark
flbr@di.ku.dk

2 Utrecht University, the Netherlands
m.loffler@uu.nl

3 Universitat Politècnica de Catalunya, Spain
rodrigo.silveira@upc.edu

Abstract

Beacon routing is the study of how to indirectly route objects through various domains. Consider two points a - which we think of as the attractor - and b - which we think of as the ball or beacon - lying on a smooth closed curve in the plane. In curve-restricted beacon routing, b moves along the curve as long as its Euclidean straight-line distance to a decreases, until this distance is locally minimal. Assuming b moves infinitely faster than a, the goal is to move a along the curve in such a way that a ends up meeting b. We say that a curve is universal if there always exists a strategy to catch the ball from every initial configuration of the attractor and the ball. Recent work of Abrahamsen et al. has shown that every simple curve is universal. The authors also conjectured that all curves with rotation number one are universal. In this note, we disprove their conjecture and present a curve with rotation number one which is not universal.

1 Introduction

Beacon routing [2] is the study of how to indirectly route objects through various domains. Let $\gamma : S^1 \to \mathbb{R}^2$ be a smooth closed curve in the plane, and let a and b be two points on γ. In curve-restricted beacon routing [1] b always moves closer to a if possible while staying on the curve. More precisely, b moves as long as its distance to a decreases, until a point where the distance is locally minimal. As soon as the position of a allows b to get closer to a, b moves again. We may think of a as the *attractor* and b as the *ball*. We can directly control the position a, but we have no direct control over b. We assume that b moves infinitely faster than a. Our goal is to move a along the curve in such a way that a meets b (see Figure 1). Note that a bit of care is required to properly define this problem and make sure γ is generic enough for the problem to be well-defined. As is carefully explained in [1, Section 3.1], we must assume that γ is regular, C^2-continuous, and intersects its evolute transversely, away from its cusps.

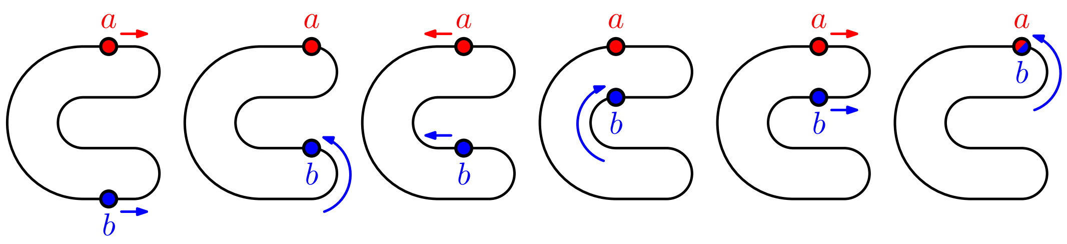


Figure 1 An example of a strategy for the attractor to catch the ball on a c-shaped closed simple curve. (Figure from [1].)

41st European Workshop on Computational Geometry, Liblice, Czech republic, April 9–11, 2025.
This is an extended abstract of a presentation given at EuroCG'25. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

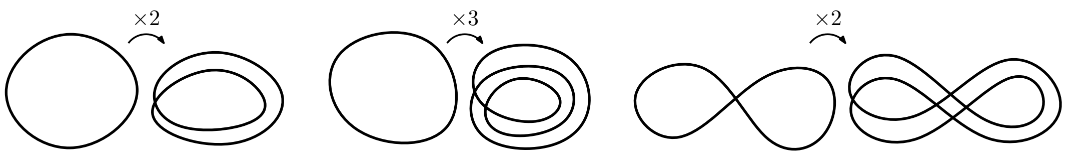


Figure 2 Any curve can by "multiplied" by taking multiple copies of it at (almost) the same location.

A given curve γ is *universal* if there always exists a strategy to catch the ball from every initial configuration [1]. Biro et al. [2] ask whether every *simple* curve is universal; Abrahamsen et al. [1] show that the answer is yes. There also exist non-universal curves; the simplest such curve is the "double circle", and in fact any curve that is "doubled" or "multiplied" is an example of such a curve, since a and b will always stay close to each other on different "copies" of the curve. Refer to Figure 2 for some examples; for proper definitions and proofs see [1].

The *rotation number* (also called the *index* or *Whitney index*) [4] of a closed curve is defined as the number of revolutions a tangent vector completes as it traverses the curve once. Inspired by the fact that by multiplying curves as in Figure 2 we can create counter examples with any rotation number $k > 1$, Abrahamsen et al. [1] conjectured that every curve with rotation number one is universal.

In this note, we disprove their conjecture by presenting a curve with rotation number 1 that is not universal.

2 Counterexample

▶ **Theorem 2.1.** *There exists a curve γ of rotation number 1 and a pair of points (a, b) on γ such that, no matter how a moves on γ, a will never reach b.*

We illustrate our construction in Figure 3. We begin by defining a family of paths $c_1, c_2, \ldots, c_7 : [0, 4] \to \mathbb{R}^2$ such that:

1. For all $i \in [7]$ and for choices of constants $C, \delta, \epsilon > 0$, we set:
   - $c_i(0) = (-1, -i\epsilon)$ and $c_i{}'(0) = (1, 0)$.
   - $c_i(1) = (1, i\epsilon + \delta)$ and $c_i{}'(1) = (1, 0)$, choosing δ so that $c_3(0) = c_3(1)$, i.e. $\delta = \epsilon(i + 1)$.
   - $c_i(2) = (1, i\epsilon - C)$ and $c_i{}'(2) = (-1, 0)$.
   - $c_i(3) = (-1, i\epsilon - C)$ and $c_i{}'(3) = (-1, 0)$.
   - $c_i(4) = c_{i+1}(0)$ and $c_i{}'(4) = (1, 0)$, with indices wrapping around so that $c_8 \coloneqq c_1$.
2. For all $t \in (0, 1) \cup (2, 3)$ and $i \in [7]$, $c_i(t)$ is contained in the strip $\{(x, y) | -1 < x < 1\}$.
3. For all $i \in [7] - \{2\}$, the tangent vector of each path c_i does not accomplish a single revolution, as t ranges from 0 to 1. The tangent vector of c_2 is the exception, and makes a single positive revolution.
4. For all $i \in [7]$, c_i traces a counter-clockwise loop between $c_i(1)$ and $c_i(2)$, so that the tangent vector of each path c_i accomplishes a single negative revolution as t ranges from 1 to 2.
5. For all $i \in [7]$, c_i traces a horizontal line segment between $c_i(2)$ and $c_i(3)$.
6. For all $i \in [7]$, c_i smoothly interpolates between $c_i(3)$ and $c_i(4)$ without adding to the rotation number.
7. For each fixed $i \in [7]$, the geometric image of the intervals $(0, 1)$, $(1, 2)$, $(2, 3)$ and $(3, 4)$ under c_i are all pairwise disjoint.

By construction, the union of such a collection of 7 paths, $\cup_i c_i$, is a closed curve γ with rotation number one. The crucial desired property that we would like to ensure with such a construction is the following:

▶ **Lemma 2.2.** *The paths $c_1, c_2, \dots, c_7$ can be chosen such that, for all $i \in [7]$, if we position a and b either respectively at starting positions $c_i(0)$ and $c_{i+1}(0)$ or starting positions $c_i(1)$ and $c_{i+1}(1)$, then no matter how we move a along c_i:*

- *If we reach a point where a is at $c_i(1)$, then b is at $c_{i+1}(1)$.*
- *If we reach a point where a is at $c_i(0)$, then b is at $c_{i+1}(0)$.*

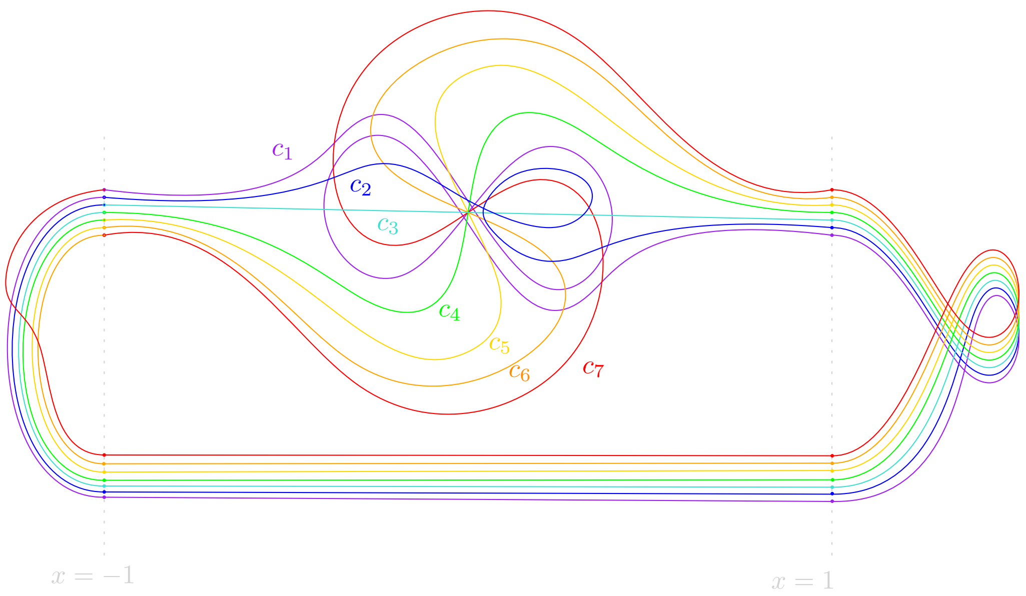


Figure 3 Our construction γ has rotation number one and is not universal: there exist configurations from which the ball can never be caught. For example, start with a at one of the lowest points of the curve (on the purple strand) and b just above it (on the blue strand). The vertically aligned points have x-coordinates either -1 or 1.

Sketch of Proof. To motivate the Lemma and our counter-example, recall that, as discussed earlier, the double circle curve is a simple example of a non-universal curve. One way to look at our construction and to think of Lemma 2.2 is to take the necessary steps to alter the double circle into a curve of rotation one, while keeping the desired non-universality. To that effect, let us picture the double circle as being obtained from two copies of a circle at almost the same location, one blue (call it c_2) and the other turquoise (call it c_3), cutting both at an arbitrary point, and switching their endpoints before reconnecting. Suppose that the outer loop ends up being the blue loop c_2. Since the double circle has rotation number two, we add a counter clockwise loop to subtract one to the rotation number, get again a curve with rotation number one (see Figure 4).

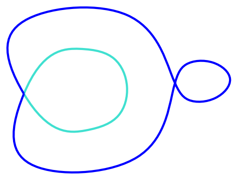


Figure 4 The doubled circle can be thought of as a prototype example with two paths (blue and turquoise). We modify the doubled circle with a counter clockwise loop (on the right) to achieve rotation number 1. However these two paths fail to verify Lemma 2.2.

Taking a closer look at this curve, we see that unfortunately this construction only works in one direction. If the attractor a is on the blue curve c_2, then b follows on the turquoise curve c_3. However if a leads on the turquoise curve, then b gets "stuck" in the blue loop we introduced, after which a need only catch up to it (see Figure 5). We show that we can introduce a sequence of 5 "intermediate" curves to solve this problem. ◀

▶ Remark. Note that the lemma is clear for all parameters $t \in [1, 4]$ so we may consider only the restriction of curves c_i to the interval $(0, 1)$.

Proof of Lemma 2.2. In light of the previous explanation and the remark, we start by defining c_2 (the blue curve on Figure 5) to trace a single counter-clockwise loop between $c_2(0)$ and $c_2(1)$ while c_3 traces a line segment between $c_3(0)$ and $c_3(1)$ (the turquoise curve on Figure 5).

▶ Remark. Note that in our analysis, for the sake of simplicity and economy of diagram making, we shall assume that a moves from the left to the right in the forward direction. For all the pairs of curves we introduce, a direct symmetry argument solves the backwards direction. The only exception is the pair c_1 and c_2, for which we detail the backwards case explicitly in .

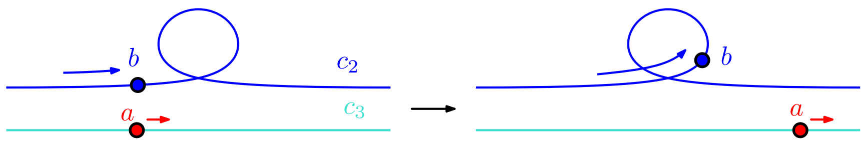


Figure 5 If the attractor a leads on the turquoise curve c_3, the ball b gets trapped in the loop on the blue curve c_2.

The first intermediate curve is the curve c_1 (in purple on Figure 6), which is essentially a multiplied copy of a curve "shaped like an 8" and thus has rotation number 0 (see the rightmost diagram of Figure 2). In the forward case, as seen on Figure 6 (top), this curve has the property that if a leads on c_1, then b follows on c_2 and escapes the added loop. We thus now have curves c_1, c_2 and c_3 with the desired property of Lemma 2.2 with the exception of the last pair c_3 and c_1. Namely, if a leads on c_3 (turquoise), b does not follow on c_1 (purple). The backwards case is illustrated on Figure 6 (bottom).

To fix this last problem, we introduce the curve c_7 (pictured in red), which is designed as a spiral to ensure that if a leads on c_7, b follows along on c_1 (purple) and is guided to escape the double eight (see Figure 7). The last issue to fix now is that if a leads on c_3 (turquoise), b will not quite follow on this new red curve c_7. However note that c_3 and c_7 have the same topology and are both a simple line with no loops. So we need only introduce a fine enough sequence of intermediate curves that interpolate between the flat line curve c_3 (turquoise) and the spiraling c_7 (red). As can be seen on the original Figure 3 depicting our counter example, three intermediate curves c_4, c_5, c_6 that each progressively spiral with less than a right-angled bend from each other are enough to accomplish this task, leading to the claim in the lemma. Note that we implictly also rely on the following remark, which should be clear from the description of our process.

▶ Remark. At any point and for all the curve pairs we've constructed, moving a back and forth instead of simply moving a monotonously along the curve does not introduce any new dynamics and we can safely assume without loss of generality that a moves monotonously along the curve it is on.

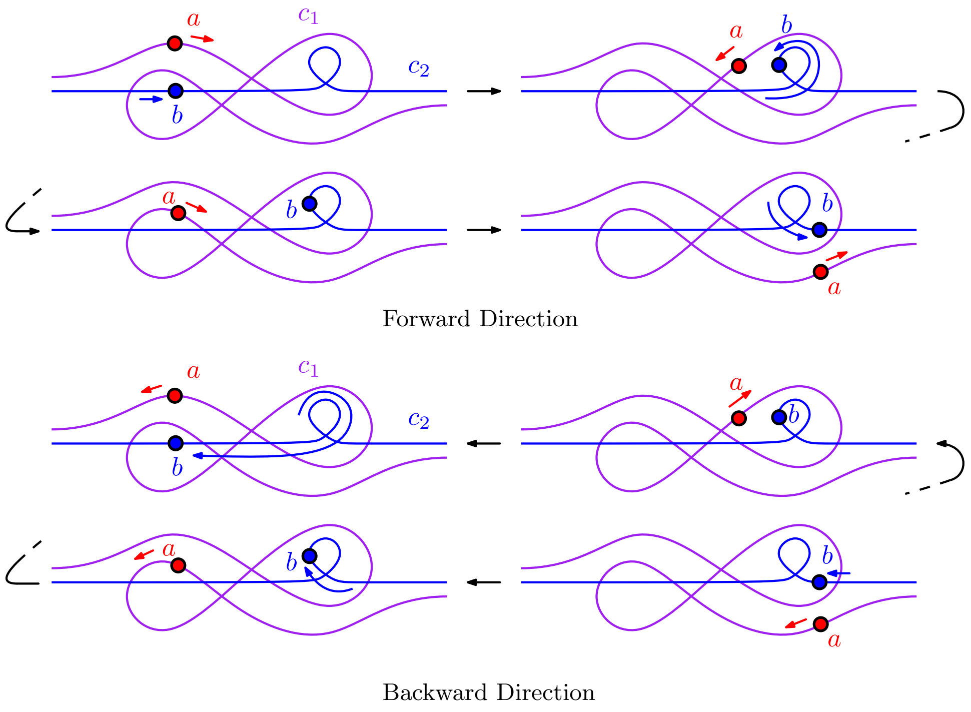


Figure 6 Adding an intermediate purple curve c_1 to act as c_3 for c_2 fixes our previous problem, where if a leads on the purple c_1, b now follows on the blue c_2 instead of getting stuck in its loop. However, we simply postponed the problem to the last pair of curves c_3 and c_1.

◀

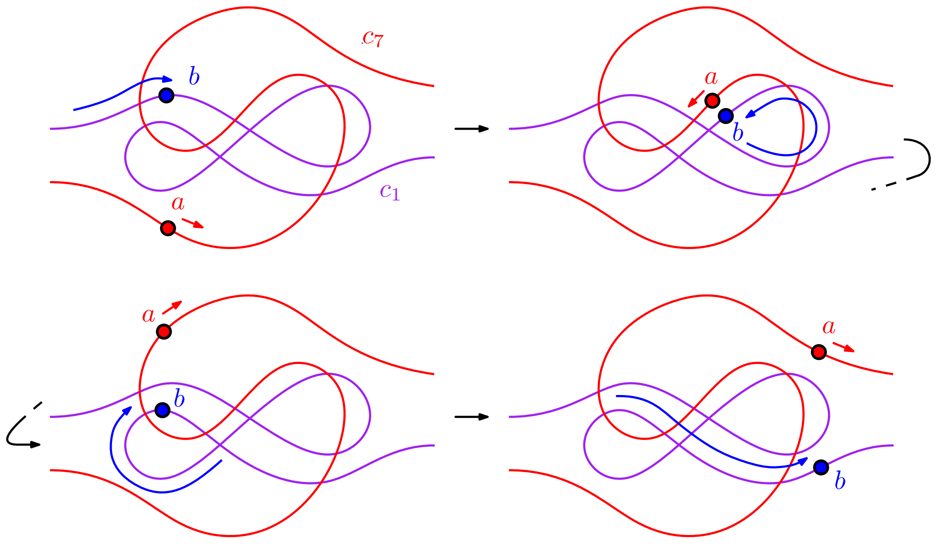


Figure 7 Introducing the red curve c_7 to act as c_3 for c_1 solves the previous pathology: if a leads on c_7, b now follows on c_1. We have thus moved the problem to finding intermediate curves from the much simpler c_7 back to the flat turquoise curve c_3.

3 Discussion

We have presented a curve that has rotation number one, but is not universal, answering in the negative the question posed by Abrahamsen et al. [1].

However, our curve is arguable not very close to being simple, since it is relatively complicated, and has 108 self-intersections. Thus, it would be interesting to find out whether a simpler example exists. Moreover, it would be interesting to understand whether universality of curves can be classified or related to other properties, such as the number of self-intersections.

Beyond closed curves, Ockenfels, Okamoto and Schnider initiate the study of the same problem when the underlying domain is a *graph* [3]. They show that every orthogonal straight-line plane graph is universal, and they conjecture the same is true for any straight-line plane graph. The same question for graphs with curves edges may also be asked.

Even more generally, it would be interesting to understand beacon routing on general embedded graphs, that are not necessarily planar.

Acknowledgements. This work was initiated at the Dagstuhl Seminar 24072: Triangulations in Geometry and Topology. We thank the organizers and the participants for the fruitful atmosphere.

References

1 Mikkel Abrahamsen, Jeff Erickson, Irina Kostitsyna, Maarten Löffler, Tillmann Miltzow, Jérôme Urhausen, Jordi Vermeulen, and Giovanni Viglietta. Chasing puppies: Mobile beacon routing on closed curves. *Journal of Computational Geometry*, 13(2):115–150, 2022.

2 Michael Biro, Jie Gao, Justin Iwerks, Irina Kostitsyna, and Joseph S. B. Mitchell. Combinatorics of beacon routing and coverage. In *Proceedings of the 25th Canadian Conference on Computational Geometry, CCCG 2013, Waterloo, Ontario, Canada, August 8-10, 2013*. Carleton University, Ottawa, Canada, 2013. URL: http://cccg.ca/proceedings/2013/papers/paper_74.pdf.

3 Johanna Ockenfels, Yoshio Okamoto, and Patrick Schnider. Chasing puppies on orthogonal straight-line plane graphs. In *Proc. 41st European Workshop on Computational Geometry, EuroCG 2025 (to appear)*.

4 Damián Wesenberg. A new formula for rotation number, 2020. URL: https://arxiv.org/abs/2010.01422, arXiv:2010.01422.