A Linear Time Algorithm for the Maximum Overlap of Two Convex Polygons Under Translation SoCG(2025)

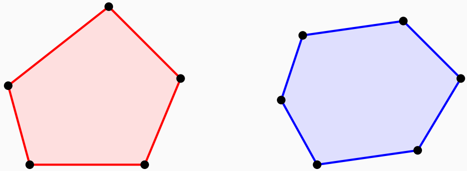
Timothy M. Chan   Isaac M. Hair

Presented by Giuseppe Pino

Introduction

Problem 1

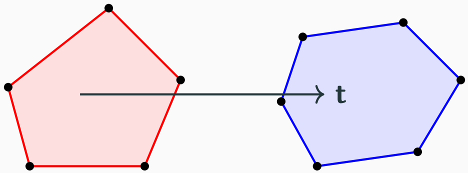
Given two convex polygons P and Q in the plane, with n and m edges respectively, find a vector in $t \in \mathbb{R}^2$ that maximizes the area of $(P + t) \cap Q$, where $P + t$ denotes P translated by t.



Motivation and Background

Problem 1

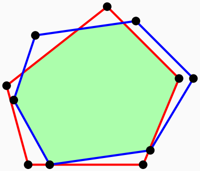
Given two convex polygons P and Q in the plane, with n and m edges respectively, find a vector in $t \in \mathbb{R}^2$ that maximizes the area of $(P + t) \cap Q$, where $P + t$ denotes P translated by t.



Motivation and Background

Problem 1

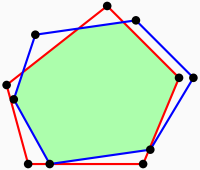
Given two convex polygons P and Q in the plane, with n and m edges respectively, find a vector in $t \in \mathbb{R}^2$ that maximizes the area of $(P + t) \cap Q$, where $P + t$ denotes P translated by t.



Motivation and Background

Problem 1

Given two convex polygons P and Q in the plane, with n and m edges respectively, find a vector in $t \in \mathbb{R}^2$ that maximizes the area of $(P + t) \cap Q$, where $P + t$ denotes P translated by t.



It can be seen as a formulation of **shape matching**: the overlap area is a natural measure of similarity between polygons.

Previous Results

Applying a multidimensional version of **parametric search** technique (1981) it is not difficult to obtain a $\mathcal{O}((n+m)\,\mathrm{polylog}(n+m))$ time algorithm.

Improvements:

- $\mathcal{O}((n+m)\log(n+m))$ by De Berg et al. (1998) using **binary search** and matrix search techniques
- Unbalanced version in $\mathcal{O}(n+m^2\log^3 n)$, which is linear for small m
- Various attempts using approximation algorithms and **prune-and-search**

The Result

A randomized (exact) algorithm for Problem 1, running in $\mathcal{O}(n+m)$ **expected time**. The first improvement in 26 years, and clearly optimal! But how?

The Result

A randomized (exact) algorithm for Problem 1, running in $\mathcal{O}(n+m)$ **expected time**. The first improvement in 26 years, and clearly optimal! But how? **Parametric search** reduces the optimization to:

- implementation of a *line oracle*
- implementation of a *point oracle* (evaluating the area of $(P+t) \cap Q$ for a given $t \in \mathbb{R}^2$).

The Result

A randomized (exact) algorithm for Problem 1, running in $\mathcal{O}(n + m)$ **expected time**. The first improvement in 26 years, and clearly optimal! But how? **Parametric search** reduces the optimization to:

- implementation of a *line oracle*
- implementation of a *point oracle* (evaluating the area of $(P + t) \cap Q$ for a given $t \in \mathbb{R}^2$).

Think of oracles as **data structures** with sublinear query time.

The Result

A randomized (exact) algorithm for Problem 1, running in $\mathcal{O}(n+m)$ **expected time**. The first improvement in 26 years, and clearly optimal! But how? **Parametric search** reduces the optimization to:

- implementation of a *line oracle*
- implementation of a *point oracle* (evaluating the area of $(P+t) \cap Q$ for a given $t \in \mathbb{R}^2$).

Think of oracles as **data structures** with sublinear query time. The clever idea is to **reduce the cost of the oracles** iteratively.

Preliminaries

Preliminaries

Lemma 1
$\sqrt{\text{Area}(P+t)\cap Q}$ *is downward concave over all values of* t *that yield nonzero overlap of* $P+t$ *with* Q*.*

Preliminaries

- Assume that P and Q have no adjacent edges and both contain the origin as an interior point.
- We order the edges in **counterclockwise** order.
- Define $N := n + m$.
- **"With high probability"** means "with probability at least $1 - N^{-\Omega(1)}$".

Prefix sums

- Let $v_1, \dots, v_n$ and $v'_1, \dots, v'_m$ be the vertices of P and Q respectively, listed in **counterclockwise** order.
- $P[v_x : v_y]$ is the **convex hull set** containing the origin, and all the vertices between v_x and v_y in counterclockwise order.

Fact: we can produce a data structure to report Area($P[v_x : v_y]$) and Area($Q(v'_x : v'_y]$) with $\mathcal{O}(N)$ **processing time** (the time needed to construct the structure) and $\mathcal{O}(1)$ **query time** (the time needed to access the wanted value).

Configuration Space

- A **polygonal chain** $[v_i : v_j]$ is the union of the consecutive edges between v_i and v_j.
- Given two edges e, e' we define the **parallelogram**

$$\pi(e, e') = \{t \in \mathbb{R}^2 \mid e + t \cap e' \neq \emptyset\}.$$

- Given two polygonal chains $A \subseteq P$ and $B \subseteq Q$ we define the set of all such line segments as

$$\Pi(A, B) = \bigcup_{e \in A, e' \in B} \partial\pi(e, e')$$

- Define $\overleftrightarrow{\Pi}$ to be $\Pi(A, B)$ where each line segment is **extended to a line**.

Angles

- **The angle** of an edge e on the boundary of an arbitrary convex polygon A, denoted $\theta_A(e)$ is the angle (counterclockwise measured) between a **horizontal rightward ray** starting at the **minimal vertex** of e, and **e**.

Angles

- **The angle** of an edge e on the boundary of an arbitrary convex polygon A, denoted $\theta_A(e)$ is the angle (counterclockwise measured) between a **horizontal rightward ray** starting at the **minimal vertex** of e, and **e**.
- For any subset X of edges of A define the **angle range** $\Lambda_A(X)$ to be the minimal interval in $[0, 2\pi)$ such that $\theta_A(e) \in \Lambda_A(X)$ for all $e \in X$.

The construction

Blocking Scheme

Given a **parameter** $b \in \mathbb{N}$, referred to as the block size, we define a partition of P and Q into blocks $P_1, \ldots, P_{\lceil N/b \rceil}$ and $Q_1, \ldots Q_{\lceil N/b \rceil}$.

First, **sort** the set

$$\{\theta_P(e) \colon e \text{ is an edge of } P\} \cup \{\theta_Q(e) \colon e \text{ is an edge of } Q\}$$

in increasing order, and then **partition** the resulting sequence into consecutive subsequences $S_1 \ldots S_{\lceil N/b \rceil}$ of length b.

Blocking Scheme

Given a **parameter** $b \in \mathbb{N}$, referred to as the block size, we define a partition of P and Q into blocks $P_1, \ldots, P_{\lceil N/b \rceil}$ and $Q_1, \ldots Q_{\lceil N/b \rceil}$.

First, **sort** the set

$$\{\theta_P(e) \colon e \text{ is an edge of } P\} \cup \{\theta_Q(e) \colon e \text{ is an edge of } Q\}$$

in increasing order, and then **partition** the resulting sequence into consecutive subsequences $S_1 \ldots S_{\lceil N/b \rceil}$ of length b.

Define P_i as the **convex hull** of the origin and set of edges of $\{e : e \text{ is an edge of } P, \text{ and } \theta_P(e) \in S_i\}$. Same for Q_j.

Blocking Scheme

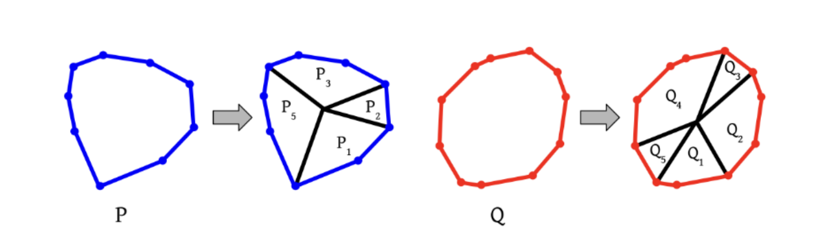


Figure 1: Blocking scheme of two polygons P and Q with $N = 20$ and $b = 4$

The Block Structure

Let $\mu \in \mathbb{R}^+$, $b \in \mathbb{N}$ and $\mathcal{T}$ be either a triangle, a line segment or a point. A $(\mu, b, \mathcal{T})$-block structure is a data structure containing the following:

1. Access to the **vertices** of P and Q in counterclockwise order with $\mathcal{O}(1)$ query time, and access to the **area prefix sums** for P and Q with $\mathcal{O}(1)$ query time.

The Block Structure

Let $\mu \in \mathbb{R}^+$, $b \in \mathbb{N}$ and $\mathcal{T}$ be either a triangle, a line segment or a point. A $(\mu, b, \mathcal{T})$-block structure is a data structure containing the following:

1. Access to the **vertices** of P and Q in counterclockwise order with $\mathcal{O}(1)$ query time, and access to the **area prefix sums** for P and Q with $\mathcal{O}(1)$ query time.
2. A block size parameter b, which implies **blocks** $P_1, \ldots, P_{\lceil N/b \rceil}$ and $Q_1, \ldots Q_{\lceil N/b \rceil}$.

The Block Structure

Let $\mu \in \mathbb{R}^+$, $b \in \mathbb{N}$ and $\mathcal{T}$ be either a triangle, a line segment or a point. A $(\mu, b, \mathcal{T})$-block structure is a data structure containing the following:

1. Access to the **vertices** of P and Q in counterclockwise order with $\mathcal{O}(1)$ query time, and access to the **area prefix sums** for P and Q with $\mathcal{O}(1)$ query time.
2. A block size parameter b, which implies **blocks** $P_1, \ldots, P_{\lceil N/b \rceil}$ and $Q_1, \ldots Q_{\lceil N/b \rceil}$.
3. A **partition** of $S = \lceil N/b \rceil$ into subsets S_{good} and S_{bad} with the requirement that $|S_{bad}| \leq \mu$.

The Block Structure

Let $\mu \in \mathbb{R}^+$, $b \in \mathbb{N}$ and $\mathcal{T}$ be either a triangle, a line segment or a point. A $(\mu, b, \mathcal{T})$-block structure is a data structure containing the following:

1. Access to the **vertices** of P and Q in counterclockwise order with $\mathcal{O}(1)$ query time, and access to the **area prefix sums** for P and Q with $\mathcal{O}(1)$ query time.
2. A block size parameter b, which implies **blocks** $P_1, \ldots, P_{\lceil N/b \rceil}$ and $Q_1, \ldots Q_{\lceil N/b \rceil}$.
3. A **partition** of $S = \lceil N/b \rceil$ into subsets S_{good} and S_{bad} with the requirement that $|S_{bad}| \leq \mu$.
4. For every $i \in S_{good}$, access in time $\mathcal{O}(1)$ to a **constant-complexity quadratic** function f_i such that $f_i(t) = \text{Area}((P_i + t) \cap Q)$ for all $t \in \mathcal{T}$.

The Idea

Problem 2 (MaxRegion)

Given a $(\mu, b, \mathcal{T})$-block structure for P and Q, find

$$\max_{t \in \mathcal{T}} \mathrm{Area}((P + t) \cap Q)$$

along with the **translation** $t^* \in \mathcal{T}$ realizing the maximum.

Cascade

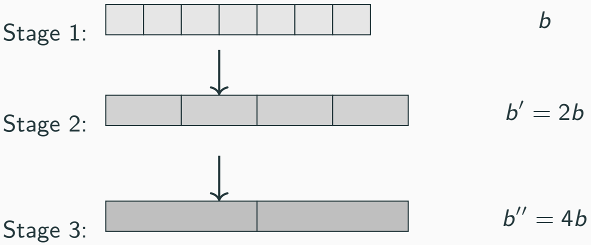
Problem 3 (Cascade)

Given a $(\mu, b, \mathcal{T})$-block structure, parameters μ' and b' such that b divides b', and triangle or line segment $\mathcal{T}' \subseteq \mathcal{T}$, either produce a $(\mu', b', \mathcal{T}')$-block structure or report failure. Failure may be reported only if the interior of $\mathcal{T}'$ intersects more than $\mu'/2$ lines from the set

$$\mathcal{S} := \bigcup_{k \in \{0,\ldots,\lceil N/b' \rceil - 1\}} \left(\bigcup_{(i,j) \in \left[\frac{b'}{b}\right] \times \left[\frac{b'}{b}\right]} \overset{\leftrightarrow}{\Pi} \left(P_{k \cdot \frac{b'}{b} + i}, Q_{k \cdot \frac{b'}{b} + j} \right) \right)$$

where $P_1, \ldots, P_{\lceil N/b \rceil}$ and $Q_1, \ldots Q_{\lceil N/b \rceil}$ are the blocks of P and Q determined by b.

Cascading process



Each cascade merges consecutive blocks ($b \to b' \to b''$), reducing their number geometrically while keeping the total work linear.

- **Fact**: $|\mathcal{S}| = \mathcal{O}(Nb')$, i.e. $\mathcal{S}$ is **near-linear** in size for all $b' = N^{o(1)}$.

- **Fact**: $|\mathcal{S}| = \mathcal{O}(Nb')$, i.e. $\mathcal{S}$ is **near-linear** in size for all $b' = N^{o(1)}$.
- Denote the **expected time complexity** of the fastest algorithm for $\textsc{MaxRegion}$ by $T_{d-\textsc{Max}}(N, b, \mu)$, where d is the dimension of the region $\mathcal{T}$.

- **Fact**: $|\mathcal{S}| = \mathcal{O}(Nb')$, i.e. $\mathcal{S}$ is **near-linear** in size for all $b' = N^{o(1)}$.
- Denote the **expected time complexity** of the fastest algorithm for $\textsc{MaxRegion}$ by $T_{d-\textsc{Max}}(N, b, \mu)$, where d is the dimension of the region $\mathcal{T}$.
- Denote the **expected time complexity** of the fastest algorithm for $\textsc{Cascade}$ by $T_{\textsc{Cascade}}(N, b, b', \mu)$.

Calculating the time complexity

Main Lemmas

Lemma 2 (Point Oracle)
For all $2 \leq b \leq N$, for all μ,

$$T_{0-\textsc{Max}}(N, b, \mu) = \mathcal{O}\left(N \cdot \frac{\log^2 b}{b} + \mu b\right).$$

Lemma 3 (Cascading Subroutine)
For all $2 \leq b, b' \leq N$ such that b divides b', for all μ,

$$T_{\textsc{Cascade}}(N, b, b', \mu) = \mathcal{O}\left(N \cdot \frac{\log b' \log b}{b} + \mu b^2\right).$$

Reducing to the Point Oracle

Theorem 4
For all $2 \leq b, b' \leq N^{o(1)}$ *such that* b *divides* b', *for all* μ, μ' *such that* $\mu' = N^{1-o(1)}$,

$$\begin{aligned} T_{2-\textsc{Max}}(N, b, \mu) =& \mathcal{O}(\log(Nb'/\mu')) \cdot T_{1-\textsc{Max}}(N, b, \mu) + \mathcal{O}(N^{0.1+o(1)}) + \\ & \mathcal{O}(1) \cdot T_{\textsc{Cascade}}(N, b, b', \mu) + T_{2-\textsc{Max}}(N, b', \mu') \end{aligned}$$

Reducing to the Point Oracle

Theorem 4
For all $2 \leq b, b' \leq N^{o(1)}$ *such that* b *divides* b', *for all* μ, μ' *such that* $\mu' = N^{1-o(1)}$,

$$T_{2-\textsc{Max}}(N, b, \mu) = \mathcal{O}(\log(Nb'/\mu')) \cdot T_{1-\textsc{Max}}(N, b, \mu) + \mathcal{O}(N^{0.1+o(1)}) + \mathcal{O}(1) \cdot T_{\textsc{Cascade}}(N, b, b', \mu) + T_{2-\textsc{Max}}(N, b', \mu')$$

$$T_{1-\textsc{Max}}(N, b, \mu) = \mathcal{O}(\log(Nb'/\mu')) \cdot T_{0-\textsc{Max}}(N, b, \mu) + \mathcal{O}(N^{0.1+o(1)}) + \mathcal{O}(1) \cdot T_{\textsc{Cascade}}(N, b, b', \mu) + T_{1-\textsc{Max}}(N, b', \mu')$$

Proof of Theorem 4

- Start with a triangle $\mathcal{T} = \mathcal{T}_0$. Produce a **sequence** $\mathcal{T}_0 \supset \mathcal{T}_1 \supset \mathcal{T}_2 \supset \cdots \supset \mathcal{T}'$ such that each triangle contains the optimal translation t^*.
- Invoke $\textsc{Cascade}$ to produce (with high probability) a $(\mu', b', \mathcal{T}')$-**block structure**.
- **Invoke** $\textsc{MaxRegion}$ on the new block structure.

Producing $\mathcal{T}'$

Fact: if an algorithm produces the desired object "with high probability", then the expected number of attempts in $\mathcal{O}(1)$.

Producing $\mathcal{T}'$

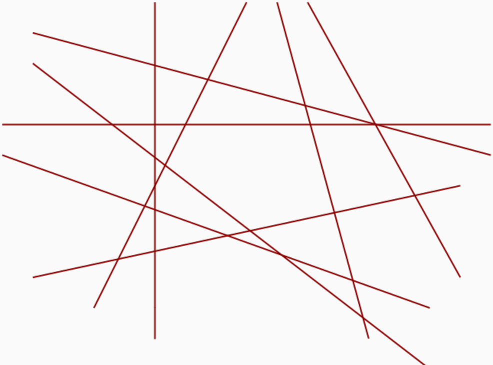
Fact: if an algorithm produces the desired object "with high probability", then the expected number of attempts in $\mathcal{O}(1)$.

Definition 5 (ε-cutting)
An ε-cutting for a set X of n hyperplanes in $\mathbb{R}^d$, for any $d \geq 1$, is a **partition** of $\mathbb{R}^d$ into simplices such that the interior of every simplex intersects at most $\varepsilon \cdot n$ hyperplanes of X.

Producing $\mathcal{T}'$

Fact: if an algorithm produces the desired object "with high probability", then the expected number of attempts in $\mathcal{O}(1)$.

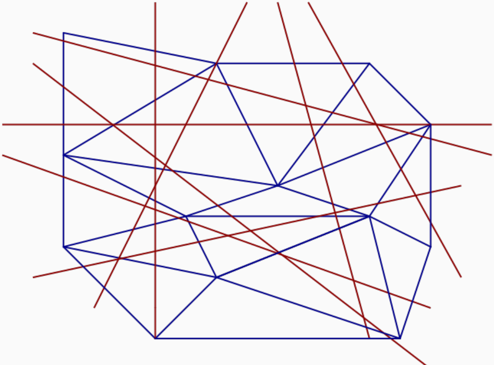
Definition 5 (ε-cutting)
An ε-cutting for a set X of n hyperplanes in $\mathbb{R}^d$, for any $d \geq 1$, is a **partition** of $\mathbb{R}^d$ into simplices such that the interior of every simplex intersects at most $\varepsilon \cdot n$ hyperplanes of X.



Producing $\mathcal{T}'$

Fact: if an algorithm produces the desired object "with high probability", then the expected number of attempts in $\mathcal{O}(1)$.

Definition 5 (ε-cutting)
An ε-cutting for a set X of n hyperplanes in $\mathbb{R}^d$, for any $d \geq 1$, is a **partition** of $\mathbb{R}^d$ into simplices such that the interior of every simplex intersects at most $\varepsilon \cdot n$ hyperplanes of X.



Lemma 6
Assume we can sample a uniformly random member of X in expected time $\mathcal{O}(n^{0.1})$. Then in $\mathcal{O}(n^{0.1+o(1)})$ expected time, we can find a set of $\mathcal{O}(1)$ simplices that constitutes an ε-cutting of X with high probability.

How to produce $\mathcal{T}_{x+1}$ given $\mathcal{T}_x$.

- With high probability produce a $\frac{1}{2}$-**cutting** for the lines of $\mathcal{S}$ that intersect the interior of $\mathcal{T}_x$.

How to produce $\mathcal{T}_{x+1}$ given $\mathcal{T}_x$.

- With high probability produce a $\frac{1}{2}$-**cutting** for the lines of $\mathcal{S}$ that intersect the interior of $\mathcal{T}_x$.
- Locate the cell of the triangulation that contains t^*. To do so, check all $\mathcal{O}(1)$ triangles:

How to produce $\mathcal{T}_{x+1}$ given $\mathcal{T}_x$.

- With high probability produce a $\frac{1}{2}$-**cutting** for the lines of $\mathcal{S}$ that intersect the interior of $\mathcal{T}_x$.
- Locate the cell of the triangulation that contains t^*. To do so, check all $\mathcal{O}(1)$ triangles:
  - For a single triangle $\mathcal{X}$, consider an **infinitesimal smaller copy** $\mathcal{X}^-$ and find the optimal translation restricted to the **edges** of $\mathcal{X}$ and $\mathcal{X}^-$.

How to produce $\mathcal{T}_{x+1}$ given $\mathcal{T}_x$.

- With high probability produce a $\frac{1}{2}$-**cutting** for the lines of $\mathcal{S}$ that intersect the interior of $\mathcal{T}_x$.
- Locate the cell of the triangulation that contains t^*. To do so, check all $\mathcal{O}(1)$ triangles:
    - For a single triangle $\mathcal{X}$, consider an **infinitesimal smaller copy** $\mathcal{X}^-$ and find the optimal translation restricted to the **edges** of $\mathcal{X}$ and $\mathcal{X}^-$.
    - If one of the line segments for $\mathcal{X}^-$ has a translation realising a larger area than all of the line segments of $\mathcal{X}$, **by concavity of the area function** $t^* \in \mathcal{X}$. Otherwise $t^* \notin \mathcal{X}$.

Proof of Theorem 4

- Start with $\mathcal{T} = \mathcal{T}_0$. Produce a **sequence** $\mathcal{T}_0 \supset \mathcal{T}_1 \supset \mathcal{T}_2 \supset \cdots \supset \mathcal{T}'$ such that each triangle contains the optimal translation t^*.

  $\Rightarrow \mathcal{O}\left(\log \frac{Nb'}{\mu'}\right) \cdot T_{\text{1-Max}}(N, b, \mu) + \mathcal{O}(N^{0.1+o(1)})$

- Invoke $\textsc{Cascade}$ to produce (with high probability) a $(\mu', b', \mathcal{T}')$-**block structure**.

  $\Rightarrow \mathcal{O}\left(\frac{N \log b' \log b}{b} + \mu b^2\right)$.

- **Invoke** $\textsc{MaxRegion}$ on the new block structure.

  $\Rightarrow T_{\text{2-Max}}(N, b', \mu')$.

Putting it all Together

Theorem 7
There is an algorithm for Problem 1 running in expected time $\mathcal{O}(n+m)$

Proof of Theorem 7

- Take $\mu := 20 \cdot N/b^3$ and $\mu' := 20 \cdot N/(b')^3$.

Proof of Theorem 7

- Take $\mu := 20 \cdot N/b^3$ and $\mu' := 20 \cdot N/(b')^3$.
- Reduce the problem to an instance of $\textsc{MaxRegion}$ over a $(20 \cdot N/2^3, 2, \mathcal{T})$-**block structure**.
- The expected running time for Problem 1 is $\mathcal{O}(N) + T_{2-\textsc{Max}}(N, 2)$.

Proof of Theorem 7

- Take $\mu := 20 \cdot N/b^3$ and $\mu' := 20 \cdot N/(b')^3$.
- Reduce the problem to an instance of $\textsc{MaxRegion}$ over a $(20 \cdot N/2^3, 2, \mathcal{T})$-**block structure**.
- The expected running time for Problem 1 is $\mathcal{O}(N) + T_{2-\textsc{Max}}(N, 2)$.
- For any $2 \leq b, b' \leq N^{o(1)}$:

$$T_{1-\textsc{Max}}(N, b) = \mathcal{O}\left(N \cdot \frac{\log b' \log^2 b}{b}\right) + T_{1-\textsc{Max}}(N, b').$$

Proof of Theorem 7

- Take $\mu := 20 \cdot N/b^3$ and $\mu' := 20 \cdot N/(b')^3$.
- Reduce the problem to an instance of $\textsc{MaxRegion}$ over a $(20 \cdot N/2^3, 2, \mathcal{T})$-**block structure**.
- The expected running time for Problem 1 is $\mathcal{O}(N) + T_{2-\textsc{Max}}(N, 2)$.
- For any $2 \leq b, b' \leq N^{o(1)}$:

$$T_{1-\textsc{Max}}(N, b) = \mathcal{O}\left(N \cdot \frac{\log b' \log^2 b}{b}\right) + T_{1-\textsc{Max}}(N, b').$$

- Solve the recursion in **two steps**: first suppose that $b \geq (logN)^{C_1}$ for some constant C_1. We will apply the following result from Toledo.

Parametric Search (Megiddo, Toledo, Cole)

- **Goal**: find the optimal parameter t^* where a concave function $F(t)$ attains its maximum.
- Assume we can **evaluate** $F(t)$ efficiently — this is the **evaluation oracle** (time T_0).
- We also have a **parallel version** of this oracle (time T_p, using P processors).
- Parametric search simulates the parallel algorithm **sequentially**, resolving comparisons that depend on t^* via oracle calls.
- **General running time** in fixed dimension d:

$$\mathcal{O}\Big(T_0(T_p \log P)^{2^d-1}\Big)$$

Parametric Search (Megiddo, Toledo, Cole)

- In our case ($d = 1$):
  - T_0 = cost of the point oracle $T_{\text{0-Max}}(N, b, \mu)$,
  - $\Rightarrow$ $T_{\text{1-Max}}(N, b) = O(N/\log N)$.

Proof of Theorem 7

- In the **second step** we solve a geometric series and setting $b' = 2b$:

$$T_{1-\textsc{Max}}\left(N \cdot \frac{\log^3 b}{b} + \frac{N}{\log N}\right)$$

Proof of Theorem 7

- In the **second step** we solve a geometric series and setting $b' = 2b$:

$$T_{1-\textsc{Max}}\left(N \cdot \frac{\log^3 b}{b} + \frac{N}{\log N}\right)$$

- Analogously, for $2 \leq b, b' \leq N^{o(1)}$:

$$T_{2-\textsc{Max}}(N, b) = \mathcal{O}\left(N \cdot \frac{\log b' \log^3 b}{b} + N \cdot \frac{\log b'}{logN}\right) + T_{2-\textsc{Max}}(N, b')$$

Proof of Theorem 7

- In the **second step** we solve a geometric series and setting $b' = 2b$:

$$T_{1-\textsc{Max}}\left(N \cdot \frac{\log^3 b}{b} + \frac{N}{\log N}\right)$$

- Analogously, for $2 \leq b, b' \leq N^{o(1)}$:

$$T_{2-\textsc{Max}}(N, b) = \mathcal{O}\left(N \cdot \frac{\log b' \log^3 b}{b} + N \cdot \frac{\log b'}{logN}\right) + T_{2-\textsc{Max}}(N, b')$$

- Again if we take $b' = 2b$:

$$T_{2-\textsc{Max}}(N, b) = \mathcal{O}\left(N \cdot \frac{\log^4 b}{b} + N \cdot \frac{\log^2 \log N}{logN}\right)$$

- $T_{2-\textsc{Max}}(N, 2) = \mathcal{O}(N)$ and the theorem follows. $\square$

Constructing the Cascading Subroutine

Lemma (Cascading Subroutine)

For all $2 \leq b, b' \leq N$ such that b divides b', for all μ,

$$T_{\textsc{Cascade}}(N, b, b', \mu) = \mathcal{O}\left(N \cdot \frac{\log b' \log b}{b} + \mu b^2\right).$$

First we need some tools to help classify the new blocks.

Lemma 8
Let X and Y be any two polygonal chains of P and Q, respectively. If $\Lambda_A(X) \cap \Lambda(Y) = \emptyset$, then X and Y intersect at most twice and we can find the intesection point(s) in time $\mathcal{O}(\log|X| \log|Y|)$.

Lemma 9
Let X and Y be polygonal chains of P and Q, respectively, and $\mathcal{T}'$ be a triangle or a line segment. If $\Lambda_P(X) \cap \Lambda_Q(Y) = \emptyset$, then we can determine if some line segment in $\Pi(X, Y)$ intersects the interior of $\mathcal{T}'$ in time $\mathcal{O}(\log|X| \log|Y|)$.

Proof of Lemma 3

- **Fact**: if $\mathcal{T}'$ is contained in a cell of $\Pi(P, Q)$, then there is a constant complexity quadratic function calculating the area.

Proof of Lemma 3

- **Fact**: if $\mathcal{T}'$ is contained in a cell of $\Pi(P, Q)$, then there is a constant complexity quadratic function calculating the area.
- Define $C_k = \{(k-1) \cdot \frac{b'}{b} + 1, \ldots, k \cdot \frac{b'}{b}\}$, i.e. $P'_k = \bigcup_{i \in C_k} P_i$ and $Q'_k = \bigcup_{j \in C_k} Q_j$. Denote by $f_k(t) = \text{Area}((P'_k + t) \cap Q'_k)$.

Proof of Lemma 3

- **Fact**: if $\mathcal{T}'$ is contained in a cell of $\Pi(P, Q)$, then there is a constant complexity quadratic function calculating the area.
- Define $C_k = \{(k-1) \cdot \frac{b'}{b} + 1, \ldots, k \cdot \frac{b'}{b}\}$, i.e. $P'_k = \bigcup_{i \in C_k} P_i$ and $Q'_k = \bigcup_{j \in C_k} Q_j$. Denote by $f_k(t) = \text{Area}((P'_k + t) \cap Q'_k)$.

$$f_k(t) = \sum_{(i,j) \in C_k \times C_k} \text{Area}\Big((P_i + t) \cap Q_j\Big) =$$

$$\sum_{i \in C_k} \text{Area}\Big((P_i + t) \cap Q_i\Big) + \sum_{\substack{(i,j) \in C_k \times C_k \\ i \neq j}} \text{Area}\Big((P_i + t) \cap Q_j\Big),$$

- Solve the two summations separately.

First Summation

- Read each function $\text{Area}((P_i + t) \cap Q_i)$ with $i \in S_{good}$. This takes $\mathcal{O}(\frac{b'}{b})$ time for each k, or $\mathcal{O}(N/b)$ for all k.

First Summation

- Read each function $\text{Area}((P_i + t) \cap Q_i)$ with $i \in S_{good}$. This takes $\mathcal{O}(\frac{b'}{b})$ time for each k, or $\mathcal{O}(N/b)$ for all k.
- For each i in S_{bad} check whether $\mathcal{T}'$ lies in a single cell of $\Pi(P_i, Q_i)$ in time $\mathcal{O}(b^2)$. If this is not the case for some i, add the index k to S_{bad}. Total time: $\mathcal{O}(\mu b^2)$.

Second Summation

- Rewrite the sum as:

$$\sum_{i\in C_k} \mathrm{Area}\Big((P_i + t) \cap Q^-_{i,k}\Big) \; + \; \sum_{i\in C_k} \mathrm{Area}\Big((P_i + t) \cap Q^+_{i,k}\Big),$$

where

$$Q^-_{i,k} \; = \; \bigcup_{\substack{j\in C_k \\ j<i}} Q_j, \qquad Q^+_{i,k} \; = \; \bigcup_{\substack{j\in C_k \\ j>i}} Q_j.$$

are **convex polygons**.

Second Summation

- Rewrite the sum as:

$$\sum_{i \in C_k} \mathrm{Area}\Big((P_i + t) \cap Q^-_{i,k}\Big) \;+\; \sum_{i \in C_k} \mathrm{Area}\Big((P_i + t) \cap Q^+_{i,k}\Big),$$

  where

$$Q^-_{i,k} \;=\; \bigcup_{\substack{j \in C_k \\ j < i}} Q_j, \qquad Q^+_{i,k} \;=\; \bigcup_{\substack{j \in C_k \\ j > i}} Q_j.$$

  are **convex polygons**.
- $\Lambda(E_P(P_i)) \cap \Lambda(E_Q(Q^-_{i,k})) = \emptyset$. So break $P_i + t$ and $Q^-_{i,k}$ into pieces that satisfy the preconditions of **Lemma** 8 and **Lemma** 9.

Second Summation

- Rewrite the sum as:

$$\sum_{i \in C_k} \text{Area}\Big((P_i + t) \cap Q^-_{i,k}\Big) \;+\; \sum_{i \in C_k} \text{Area}\Big((P_i + t) \cap Q^+_{i,k}\Big),$$

  where

$$Q^-_{i,k} \;=\; \bigcup_{\substack{j \in C_k \\ j < i}} Q_j, \qquad Q^+_{i,k} \;=\; \bigcup_{\substack{j \in C_k \\ j > i}} Q_j.$$

  are **convex polygons**.
- $\Lambda(E_P(P_i)) \cap \Lambda(E_Q(Q^-_{i,k})) = \emptyset$. So break $P_i + t$ and $Q^-_{i,k}$ into pieces that satisfy the preconditions of **Lemma** 8 and **Lemma** 9.
- In time $\mathcal{O}(\log b \log b')$ we determine if $\mathcal{T}'$ **lies in a single cell** of $\Pi(P_i, Q^-_{i,k})$. If it happens, calculate the functions. Otherwise add k to S_{bad}.

Conclusion of the proof

- By summing over i and k gives **total time** $\mathcal{O}(N \cdot \frac{\log b \log b'}{b})$.

Conclusion of the proof

- By summing over i and k gives **total time** $\mathcal{O}(N \cdot \frac{\log b \log b'}{b})$.
- Each line in $\mathcal{S}$ corresponds to **at most two** k's in S_{bad}.

Conclusion of the proof

- By summing over i and k gives **total time** $\mathcal{O}(N \cdot \frac{\log b \log b'}{b})$.
- Each line in $\mathcal{S}$ corresponds to **at most two** k's in S_{bad}.
- Thus if $|S_{bad}| > \mu$, then more than $\mu/2$ lines of $\mathcal{S}$ intersect $\mathcal{T}'$. $\square$

Thank you for the attention