A generalization of AT-free graphs and a generic algorithm for solving triangulation problems

Hajo Broersma
Faculty of Applied Mathematics
University of Twente, P.O. Box 217
7500 AE Enschede, the Netherlands
H.J.Broersma@math.utwente.nl

Ton Kloks
Department of Applied Mathematics
Charles University
Prague
kloks@kam.ms.mff.cuni.cz

Dieter Kratsch Haiko Müller
Fakultät für Mathematik und Informatik
Friedrich-Schiller-Universität Jena
07740 Jena, Germany
{kratsch,hm}@minet.uni-jena.de

Abstract

A subset A of the vertices of a graph G is an *asteroidal set* if for each vertex $a \in A$, the set $A \setminus \{a\}$ is contained in one component of $G - N[a]$. An asteroidal set of cardinality three is called *asteroidal triple* and graphs without an asteroidal triple are called *AT-free*. The maximum cardinality of an asteroidal set of G, denoted by $\mathsf{an}(G)$, is said to be the *asteroidal number* of G. We present a scheme for designing algorithms for triangulation problems on graphs. As a consequence, we obtain algorithms to compute graph parameters such as treewidth, minimum fill-in and vertex ranking number. The running

time of these algorithms is a polynomial (of degree asteroidal number plus a small constant) in the number of vertices and the number of minimal separators of the input graph.

1 Introduction

Graphs without an asteroidal triple are called asteroidal triple-free graphs (short AT-free graphs) and attained much attention recently. Möhring has shown that every minimal triangulation of an AT-free graph is an interval graph, which implies that for every AT-free graph the treewidth and the pathwidth of the graph are equal [24]. Furthermore a collection of interesting structural and algorithmic properties of AT-free graphs has been obtained by Corneil, Olariu and Stewart, among them an existence theorem for so-called dominating pairs in connected AT-free graphs and a linear time algorithm to compute a dominating pair for connected AT-free graphs (see [10, 11]).

The class of graphs with bounded asteroidal number extends the class of AT-free graphs, based on a natural way of generalizing the concept of asteroidal triples to so-called asteroidal sets, first given by Walter [27]. Walter, Prisner and Lin et al. used asteroidal sets to characterize certain subclasses of the class of chordal graphs [22, 25, 27].

In this paper we consider the NP-complete graph problems TREEWIDTH, MINIMUM FILL-IN and VERTEX RANKING that all remain NP-complete when restricted to AT-free graphs. In fact, each of the three problems remains NP-complete on cobipartite graphs [2, 6, 28], that form a small subclass of the class of AT-free graphs.

TREEWIDTH has been studied in numerous recent papers, mainly since many NP-complete graph problems become solvable in polynomial time or even linear time when restricted to the class of graphs with bounded treewidth [1, 4, 15]. Recall that for each constant k, there is a linear time algorithm that determines whether a given graph has treewidth at most k [5, 15]. However, the constant factor of this algorithm is exponential in the treewidth (of yes-instances), which limits its practicality.

The MINIMUM FILL-IN problem stems from the optimal performance of Gaussian elimination on sparse matrices and has important applications in this area. Both TREEWIDTH and MINIMUM FILL-IN ask for a certain chordal embedding of the given graph. This often allows the design of similar algorithms for both problems, when graphs of some special class are considered.

The VERTEX RANKING problem received much attention lately because of the growing number of applications. The problem of finding an optimal vertex ranking is equivalent to the problem of finding a minimum-height elimination tree of a graph [12]. This measure is of importance for the parallel Cholesky factorization of matrices [7, 23]. Other applications can be found in VLSI-layout design [20].

Using an algorithm of [16] to list all minimal separators of a given graph G in time $O(n^5 r + m)$, where n is the number of vertices of G, m is the number of edges of G, and r is the number of minimal separators of G, one has designed algorithms with a running time bounded by a polynomial in the number of vertices and the number of minimal separators of the input graph, that compute the treewidth and the minimum fill-in [18] as well as the vertex ranking number [19] on AT-free graphs. Notice that the running time of these algorithms is not bounded by a polynomial in the input length, since AT-free graphs may have 'exponentially' many minimal separators (see, e.g., [19] for an example of this).

We generalize the method used in [18, 19]. To be more precise, we focus on certain sets of minimal separators called blocking sets. We show that these blocking sets have at most $\mathsf{an}(G)$ elements, and that they decompose the graph into a number of so-called blocks, which is bounded by a polynomial of order $\mathsf{an}(G)$ in the number of minimal separators of G. We consider graphs H obtained from a block of G by making the separators of the blocking set complete, and establish a relation between the blocks of H and the blocks of G. Together with some known recurrence relations for the three aforementioned problems in terms of the minimal separators S of G and the components of $G - S$, this enables us to give a scheme for recursive algorithms. In this way, for each of the three problems, we obtain an algorithm that solves the corresponding problem for all graphs G in time $O(n^5 r + m + kr^{k+1}(n+m)n\log n)$, where $k0\mathsf{an}(G)$ and r is the number of minimal separators of G. Moreover, the algorithms can be implemented without knowing the asteroidal number or the number of minimal separators of the input graphs in advance. In that case, the algorithms will generate the correct answers, within the stated timebound. This is of importance, since computing the asteroidal number in general is NP-complete [17].

Our results are related to an interesting open problem concerning Treewidth and Minimum fill-in: Does there exist an algorithm to compute the treewidth (resp. the minimum fill-in) of any graph in time bounded by a polynomial in the number of vertices and the number of minimal separators of the given graph?

2 Preliminaries

Throughout the paper, we use $G = (V, E)$ to denote a graph with vertex set V and edge set E, and we let $|V| = n$ and $|E| = m$. For $W \subseteq V$, $G[W]$ denotes the subgraph of G induced by the vertices of W, $G - W$ is shorthand for $G[V \setminus W]$. For a vertex $x \in V$, we write $G - x$ instead of $G - \{x\}$. The set of vertices adjacent to a vertex $x \in V$ is the *neighborhood* $N(x)$ of x, and $N[x] = \{x\} \cup N(x)$ is the *closed neighborhood* of x. We say that a sequence $P = (u_0, u_1, \dots, u_l)$ of pairwise distinct vertices of G is a u, v-path in G if $u = u_0$, $v = u_l$, and $\{u_{i-1}, u_i\} \in E$ for $i = 1, \dots, l$. For any set $\mathfrak{S}$ whose elements are sets itself we use $\bigcup \mathfrak{S}$ to denote $\bigcup_{S \in \mathfrak{S}} S$.

2.1 Preliminaries on asteroidal sets

Definition 1. A subset $A \subseteq V$ is called an *asteroidal set* of G if for each $a \in A$ the vertices of $A \setminus \{a\}$ are contained in one component of $G - N[a]$. The maximum cardinality of an asteroidal set of G is denoted by $\mathsf{an}(G)$, and is called the *asteroidal number* of G.

By definition the vertices of an asteroidal set are pairwise nonadjacent. Hence $\mathsf{an}(G) \le \alpha(G)$, where $\alpha(G)$ denotes the maximum cardinality of an independent set in G. Furthermore for every positive integer k there exist graphs with asteroidal number k, e.g. $\mathsf{an}(C_{2k}) = k$ for $k \ge 2$, where C_n is the chordless cycle on n vertices. Notice that every subset of an asteroidal set is itself asteroidal.

There are polynomial time algorithms to compute the asteroidal number for graphs in some special classes, like HHD-free graphs (including all chordal graphs), claw-free graphs, circular-arc graphs and circular permutation graphs. However the corresponding decision problem remains NP-complete on triangle-free 3-connected 3-regular planar graphs [17].

An asteroidal set of cardinality three was called an asteroidal triple (short AT) in [21], where it was shown that chordal graphs without AT are exactly those that are interval graphs.

2.2 Preliminaries on triangulations

Definition 2. A graph H is *chordal* (or *triangulated*) if it does not contain a chordless cycle of length at least four as an induced subgraph.

Definition 3. A *triangulation* of G is a graph H with the same vertex set as G such that H is chordal and G is a subgraph of H. A triangulation H

of G is called *minimal* if there is no proper subgraph H' of H which is also a triangulation of G.

The following theorem was proved in [26]. Here $H \dot{-} e$ denotes the graph obtained from H by removing the edge e.

Theorem 4. *Let H be a triangulation of a graph G. The graph H is a minimal triangulation of G if and only if for every edge $e \in E(H) \setminus E(G)$ the graph $H \dot{-} e$ is not chordal. Hence every edge $e \in E(H) \setminus E(G)$ is the unique chord in a cycle of length four in H.*

The size of a maximum clique in G is denoted by $\omega(G)$.

Definition 5. The *treewidth* of G, denoted by $\mathsf{tw}(G)$, is the minimum of $\omega(H) - 1$ taken over all triangulations H of G.

Definition 6. The *minimum fill-in* of G, denoted by $\mathsf{mfi}(G)$, is the minimum of $|E(H) \setminus E|$ taken over all triangulations H of G.

Definition 7. Let t be an integer. A *(vertex) t-ranking* of G is a coloring $c : V \to \{1, \dots, t\}$ such that for every pair of vertices x and y with $c(x) = c(y)$ and for every path between x and y there is a vertex z on this path with $c(z) > c(x)$. The *vertex ranking number* of G, denoted by $\chi_{\mathrm{r}}(G)$, is the smallest value t for which the graph G admits a t-ranking.

2.3 Preliminaries on minimal separators

A proper subset $S \subset V$ is a *separator* of G if $G - S$ is disconnected.

Definition 8. A vertex set $S \subset V$ is an *a, b-separator* of G if the removal of S separates a and b in distinct components of $G - S$. If no proper subset of an a, b-separator S is an a, b-separator, then S is a *minimal a, b-separator*. A vertex set $S \subset V$ is a *minimal separator* of G if there exist nonadjacent vertices a and b in G such that S is a minimal a, b-separator of G.

We define $\mathsf{Comp}(G) = \{X : \varnothing \neq X \subseteq V \text{ and } G[X] \text{ is a component of } G\}$. By $\mathsf{Sep}(G)$ we denote the set of all minimal separators of G. The following lemma enables the design of a linear time algorithm that decides whether a given vertex set S is a minimal separator of a given graph G.

Definition 9. Let S be a separator of G. A component C of $G - S$ is *full* (w.r.t. S) if every vertex of S has at least one neighbor in C.

Lemma 10. *A set S of vertices of G is a minimal separator of G if and only if $G - S$ has at least two full components.*

Dirac established the following characterization of chordal graphs [13].

Theorem 11. *G is a chordal graph if and only if every minimal separator of G is a clique.*

For any set S, we denote by $S^{[2]}$ the set of all subsets of S of cardinality 2.

Definition 12. Let $\mathfrak{S}$ be any set of vertex subsets of G. Then $G_{\mathfrak{S}} = (V, E \cup \bigcup_{S \in \mathfrak{S}} S^{[2]})$ is the graph obtained from G by adding exactly those edges, which are not present in G and which are edges of a complete graph on some $S \in \mathfrak{S}$.

Now we can state the following characterization of minimal triangulations.

Theorem 13. *A graph H is a minimal triangulation of G if and only if $H = G_{\mathsf{Sep}(H)}$.*

Proof. Assume H is a minimal triangulation of G, and let $\mathfrak{S} = \mathsf{Sep}(H)$. Then clearly $G_{\mathfrak{S}}$ is a subgraph of H since every minimal separator in H is a clique. Let $e = \{a, b\}$ be an edge of H which is not an edge of G. Since H is a minimal triangulation, e is the unique chord of a 4-cycle (a, p, b, q) in H (Lemma 4). But then a and b must be contained in every minimal p, q-separator in H, which shows that e is also an edge in $G_{\mathfrak{S}}$.

Now assume that $H = G_{\mathfrak{S}}$. Then H is a triangulation of G since every minimal separator is a clique by Lemma 11. Let $e = \{a, b\}$ be an edge of H which is not an edge of G. Since e is an edge in $G_{\mathfrak{S}}$, e is contained in a minimal separator S of H. Let $H[C_1]$ and $H[C_2]$ be two full components of $H - S$ (Lemma 10). Then we can construct two chordless a, b-paths in H, with internal vertices in C_1 and C_2 respectively thus obtaining a chordless cycle of length at least four in $H \dot{-} e$. Hence $H \dot{-} e$ is not chordal. $\square$

Finally we mention two useful characteristics of minimal triangulations (see e.g. [18]).

Lemma 14. *If H is a minimal triangulation of G, then*

1. *If a and b are nonadjacent in H, then every minimal a, b-separator in H is also a minimal a, b-separator in G.*

2. *If S is a minimal separator in H and if C is the vertex set of a component of $H - S$, then C induces also a component in $G - S$.*

3 Recurrence relations and minimal separators

Some well-known graph parameters can be computed by applying recurrence relations involving the set of all minimal separators of the graph under consideration. The most prominent examples are the following.

First let us consider the treewidth of G. If G is not a complete graph, then G contains a minimal separator S such that $\mathsf{tw}(G) = \mathsf{tw}(G_{\{S\}})$. This leads to the following theorem shown in [18]. Here and in the sequel, $G(\{S\}, C) = G_{\{S\}}[S \cup C]$.

Theorem 15. *Let G be a graph which is not complete. Then*

$$\mathsf{tw}(G) = \min_{S \in \mathsf{Sep}(G)} \max_{C \in \mathsf{Comp}(G-S)} \mathsf{tw}(G(\{S\}, C)).$$

Now we consider the minimum fill-in of G. For all $W \subseteq V$ we define $\mathsf{fill}(W) = |W^{[2]}| - |E(G[W])|$ to be the number of edges not in E that have to be added to $G[W]$ for making W a clique. The following theorem is given in [18].

Theorem 16. *Let G be a graph which is not complete. Then*

$$\mathsf{mfi}(G) = \min_{S \in \mathsf{Sep}(G)} \Big(\mathsf{fill}(S) + \sum_{C \in \mathsf{Comp}(G-S)} \big(\mathsf{mfi}(G(\{S\}, C)) - \mathsf{fill}(S)\big)\Big).$$

Finally we consider the vertex ranking problem. Notice that $\chi_r(G) = n$ for any complete graph G on n vertices. In the following we mention a special case of a theorem, given in [12], which is sufficient for our purposes.

Theorem 17. *Let G be a graph which is not complete. Then*

$$\chi_{\mathrm{r}}(G) = \min_{S \in \mathsf{Sep}(G)} \big(|S| + \max_{C \in \mathsf{Comp}(G-S)} \chi_{\mathrm{r}}(G[C])\big).$$

We have seen that for three well-known and well-studied NP-complete graph problems one has established recurrence relations that are all of the same type. It is natural that one has tried to design efficient algorithms for special graph classes exploiting these recurrence relations. Besides many efficient algorithms on special graph classes for the three problems, one has obtained algorithms for AT-free graphs, that are based on the abovementioned recurrence relations, in [18] and [19]. A similar approach (using closed neighborhoods instead of minimal separators) leads to polynomial time algorithms for INDEPENDENT SET, INDEPENDENT DOMINATING SET and INDEPENDENT PERFECT DOMINATING SET [9].

Our major goal in the remainder of this paper is to generalize the approach for AT-free graphs to obtain a general scheme for designing recursive algorithms on all ggraphs, which is applicable as soon as there is a recurrence relation for computing the graph parameter under consideration similar to those in Theorems 15, 16 and 17.

4 Blocks

Blocking sets and blocks are central concepts for the recursive algorithms and the corresponding decompositions.

Definition 18. A set $\mathfrak{S}$ of minimal separators of G is a *blocking set* if the elements of $\mathfrak{S}$ are incomparable with respect to set inclusion and for all $S \in \mathfrak{S}$ the vertex set $(\bigcup \mathfrak{S}) \setminus S$ is contained in one component of $G - S$.

Note that in particular any singleton consisting of a minimal separator of G is a blocking set.

Definition 19. Let $\mathfrak{S}$ be a blocking set of G with $|\mathfrak{S}| \geq 2$. Then a vertex $v \in V \setminus \bigcup \mathfrak{S}$ is said to be *in the interior of* $\mathfrak{S}$ if, for every $S \in \mathfrak{S}$, the vertex v and the vertex set $\bigcup \mathfrak{S} \setminus S$ are contained in one component of $G - S$.

Lemma 20. *For every blocking set $\mathfrak{S}$ of G, $|\mathfrak{S}| \leq \mathsf{an}(G)$.*

Proof. Any asteroidal set A of G with $|A| \geq 3$ is contained in one component of G. Hence we may assume that G is connected and $|\mathfrak{S}| \geq 2$. For every minimal separator $S \in \mathfrak{S}$ of G, we choose a vertex $b(S)$ that belongs to $\bigcup \mathfrak{S} \setminus S$, and a vertex $a(S)$ in a full component of $G - S$ that does not contain the vertex set $\bigcup \mathfrak{S} \setminus S$. Thus $a(S)$ and $b(S)$ belong to different components of $G - S$ for every $S \in \mathfrak{S}$. Moreover, $a(S) \neq a(S')$ for all distinct $S, S' \in \mathfrak{S}$. We claim that $A = \{a(S) : S \in \mathfrak{S}\}$ is an asteroidal set of G, thus proving that $|A| = |\mathfrak{S}| \leq \mathsf{an}(G)$.

Let $S \in \mathfrak{S}$. The vertices of the component of $G - S$ containing $b(S)$ are in one component of $G - N[a(S)]$, say $G[C]$. The set C contains the vertex $b(S)$ and it also contains the vertex $a(S')$ for all $S' \in \mathfrak{S} \setminus \{S\}$ since there is a $b(S), a(S')$-path inside $G[C]$. To see this, note that there is a path from $b(S)$ to a vertex $s' \in S' \setminus S$ inside the component of $G - S$ containing $\bigcup \mathfrak{S} \setminus S$. Since the component of $G - S'$ containing $a(S')$ is full, there is a path from s' to $a(S')$ with all internal vertices inside the component of $G - S'$ containing $a(S')$.

Therefore for every $S \in \mathfrak{S}$, the vertex set $A \setminus \{a(S)\}$ is contained in one component of $G - N[a(S)]$, which implies that $A = \{a(S) : S \in \mathfrak{S}\}$ is indeed an asteroidal set of G. $\square$

Definition 21. A pair $(\mathfrak{S}, C)$ is a *block* of G if $\mathfrak{S}$ is a blocking set of G, $C \subseteq V$ and one of the following conditions is fulfilled.

- $|\mathfrak{S}| \geq 2$ and C is the set of all vertices in the interior of $\mathfrak{S}$.
- If $\mathfrak{S}$ contains exactly one element S, then C is the vertex set of a component of $G - S$ or $C = \varnothing$.
- $\mathfrak{S} = \varnothing$ and C is the vertex set of a component of G.

The definition and Lemma 20 immediately imply

Observation 22. *The number of different blocks of G is at most*

$$(|\mathsf{Sep}(G)| + 1) \cdot |V| + \sum_{k=2}^{\mathsf{an}(G)} \binom{|\mathsf{Sep}(G)|}{k}.$$

(Notice that $\binom{n}{k} = 0$ if k and n are integers with $0 \leq n < k$.)

The following definition is motivated by the recurrence relations in Section 3 and Theorems 11 and 13.

Definition 23. The *realization* $G(\mathfrak{S}, C)$ of a block $(\mathfrak{S}, C)$ of G is the graph $G_{\mathfrak{S}}[C \cup \bigcup \mathfrak{S}]$.

The definition implies that the realization of any block is a connected graph.

5 Decomposing blocks

We consider a block $(\mathfrak{S}, C)$ of G, its realization $H = G(\mathfrak{S}, C)$ and a minimal separator T of H. Then for an arbitrary component $H[D]$ of $H - T$, the pair $(\{T\}, D)$ is a block of H. Our major goal in this section is to prove that any block $(\{T\}, D)$ of H can be described as a block of G in the following sense: For any block $(\{T\}, D)$ of $H = G(\mathfrak{S}, C)$, there is a block $(\mathfrak{T}, D')$ of G such that the corresponding realizations are exactly the same graphs, i.e., $G(\mathfrak{T}, D') = H(\{T\}, D)$.

The consequence is that any algorithm, which recursively computes a minimal separator T for the current graph H and then calls itself on the realization of the block $(\{T\}, D)$ for each component D of $H - T$ until the current graph is complete, will only work on realizations of blocks of the input graph G. Together with Lemma 20 and Observation 22 this implies, that each recursive algorithm of this type checks at most $O(|\mathsf{Sep}(G)|^{\mathsf{an}(G)})$ realizations of blocks of the input graph G.

We start with a lemma that is essential for this section.

Lemma 24. *Let $(\mathfrak{S}, C)$ be a block of G and let x and z be distinct vertices in $C \cup \bigcup\mathfrak{S}$. Then $\{x, z\}$ is an edge of $G(\mathfrak{S}, C)$ if and only if $\{x, z\} \in E$ or there exists an integer $l \geq 1$ and a path $(x, y_1, y_2, \ldots, y_l, z)$ in G with $y_i \in V \setminus (C \cup \bigcup\mathfrak{S})$ for $i = 1, \ldots, l$.*

Proof. First let $\{x, z\}$ be an edge of $G(\mathfrak{S}, C)$ and suppose $\{x, z\} \notin E$. Then there is a minimal separator $S \in \mathfrak{S}$ such that $x, z \in S$. Since $(\mathfrak{S}, C)$ is a block of G there is a full component $G[D]$ of $G - S$ with $D \cap (C \cup \bigcup\mathfrak{S}) = \varnothing$ by Lemma 10. We choose vertices x' and z' in D such that $\{x, x'\}, \{z, z'\} \in E$. Since $G[D]$ is connected there exists a path $(y_1, y_2, \ldots, y_l)$ in $G[D]$ with $y_1 = x'$ and $y_l = z'$, $l \geq 1$. Consequently, $(x, y_1, y_2, \ldots, y_l, z)$ is a path in G with $y_i \in V \setminus (C \cup \bigcup\mathfrak{S})$ for $i = 1, \ldots, l$.

Now suppose $\{x, z\} \notin E$. Let $l \geq 1$ be an integer and $(x, y_1, y_2, \ldots, y_l, z)$ be a path in G with $y_i \in V \setminus (C \cup \bigcup\mathfrak{S})$ for $i = 1, \ldots, l$. Then there is a minimal separator $S \in \mathfrak{S}$ such that $x, z \in S$ and $\{x, z\}$ is an edge of the realization $G(\mathfrak{S}, C)$. □

Now we consider a path $(x_1, x_2, \ldots, x_l)$ in the realization $G(\mathfrak{S}, C)$ of a block $(\mathfrak{S}, C)$ of G. By Lemma 24 we are able to insert vertices from $V \setminus (C \cup \bigcup\mathfrak{S})$ into the sequence $(x_1, x_2, \ldots, x_l)$ such that the resulting sequence is a path in G. On the other hand, consider a x, z-path in G with $x, z \in C \cup \bigcup\mathfrak{S}$. Then by Lemma 24 we obtain a x, z-path in $G(\mathfrak{S}, C)$ if we remove all vertices outside $C \cup \bigcup\mathfrak{S}$ from the sequence $(x, \ldots, z)$. This observation proves the next lemma.

Lemma 25. *Let $(\mathfrak{S}, C)$ be a block of G with realization $H = G(\mathfrak{S}, C)$, let $T \subset C \cup \bigcup\mathfrak{S}$, and let x and z be distinct vertices in $(C \cup \bigcup\mathfrak{S}) \setminus T$. Then x and z are in one component of $H - T$ if and only if x and z are in one component of $G - T$. Particularly, for every minimal separator $S \in \mathfrak{S}$ the set $S \setminus T$ is contained in one component of $G - T$.*

Proof. By Lemma 24. □

Note that the last two lemmas are close to Lemma 14. Next we are interested in minimal separators of realizations.

Lemma 26. *Let $(\mathfrak{S}, C)$ be a block of G and let a and b be nonadjacent vertices in $G(\mathfrak{S}, C)$. Then every minimal a, b-separator in $G(\mathfrak{S}, C)$ is a minimal a, b-separator in G.*

Proof. Let T be a minimal a, b-separator in $H = G(\mathfrak{S}, C)$. Then by Lemma 25 the set T is an a, b-separator in G.

Let $G[C_a]$ and $G[C_b]$ be the components of $G - T$ containing a and b, and let $H[D_a]$ and $H[D_b]$ be the components of $H - T$ containing a and b, respectively. We apply Lemma 10. Every vertex of T has in H a neighbor in D_a and a neighbor in D_b. By Lemma 24 every vertex of T has in G a neighbor in C_a and a neighbor in C_b. Hence T is a minimal a, b-separator of G. □

The next lemma classifies the minimal separators of realizations into three types.

Lemma 27. *Let $(\mathfrak{S}, C)$ be a block of G and let T be a minimal separator of $H = G(\mathfrak{S}, C)$. Then exactly one of the following three conditions holds:*

Type 1: *there are distinct minimal separators $S_1, S_2 \in \mathfrak{S}$ with $T \subset S_1$ and $T \subset S_2$,*

Type 2: *there is exactly one minimal separator $S_0 \in \mathfrak{S}$ such that $T \subset S_0$,*

Type 3: *$T \setminus S \neq \varnothing$ for all $S \in \mathfrak{S}$.*

Furthermore, in Types 1 and 2 the graph $H - T$ has exactly two components.

Proof. First let $S \in \mathfrak{S}$ be a minimal separator with $T \subseteq S$. Then $T \neq S$ since $\mathfrak{S}$ is a blocking set. The graph $H[S \setminus T]$ is a component of $H - T$ by Lemma 25. Another component is induced by $C \cup \bigcup \mathfrak{S} \setminus S$ since $(\mathfrak{S}, C)$ is a block and again by Lemma 25. Consequently, there is no third component of $H - T$ if T is of type 1 or 2. □

Proposition 28 (Type 1). *Let $(\mathfrak{S}, C)$ be a block of G and let T be a minimal separator of $H = G(\mathfrak{S}, C)$ such that there exist at least two distinct minimal separators in $\mathfrak{S}$ containing T. Then $C = \varnothing$, $|\mathfrak{S}| = 2$ and for each $S \in \mathfrak{S}$ we have $H(\{T\}, S \setminus T) = G(\{S\}, \varnothing)$.*

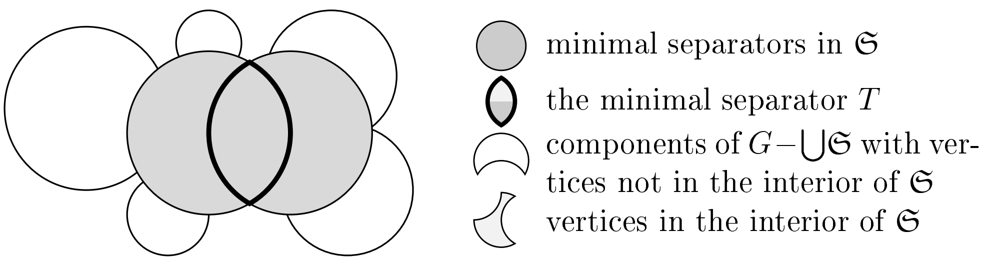

Figure 1: Type 1

Proof. Let $S_1, S_2 \in \mathfrak{S}$ be distinct minimal separators with $T \subseteq S_1 \cap S_2$. By Lemma 27 the components of $H - T$ are induced by $S_i \setminus T$ and $C \cup \bigcup \mathfrak{S} \setminus S_i$, both for $i = 1$ and for $i = 2$. Since $S_1 \setminus T \neq S_2 \setminus T$ we have $C = \varnothing$, $\mathfrak{S} = \{S_1, S_2\}$ and $T = S_1 \cap S_2$. Now the statement for the realization is obvious. $\square$

Let $(\{S_1, S_2\}, \varnothing)$ be a block of G. By Proposition 28 the unique minimal separator $T = S_1 \cap S_2$ of $G(\{S_1, S_2\}, \varnothing)$ decomposes $(\{S_1, S_2\}, \varnothing)$ into two other blocks of G. We define the decomposition of $(\{S_1, S_2\}, \varnothing)$ by

$$\mathsf{Dec}(\{S_1, S_2\}, \varnothing, T) = \{(\{S_1\}, \varnothing), (\{S_2\}, \varnothing)\}.$$

Proposition 29 (Type 2). *Let $(\mathfrak{S}, C)$ be a block of G and let T be a minimal separator of $H = G(\mathfrak{S}, C)$ such that there is a unique minimal separator $S_0 \in \mathfrak{S}$ with $T \subset S_0$. Let $\mathfrak{T} = \mathfrak{S} \setminus \{S_0\}$ and $D = C \cup \bigcup \mathfrak{T}$. Then $H[D]$ and $H[S_0 \setminus T]$ are the components of $H - T$. Furthermore $(\{T\} \cup \mathfrak{T}, C)$ is a block of G with $G(\{T\} \cup \mathfrak{T}, C) = H(\{T\}, D)$, and $(\{S_0\}, \varnothing)$ is a block of G with $G(\{S_0\}, \varnothing) = H(\{T\}, S_0 \setminus T)$.*

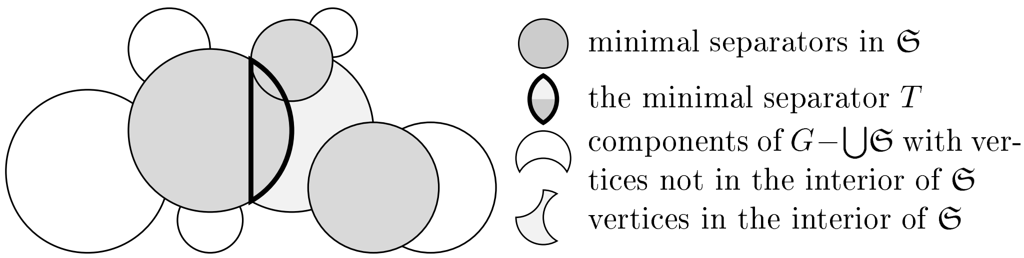

Figure 2: Type 2

Proof. By Lemma 27 the graph $H - T$ has exactly two components. These are $H[D]$ and $H[S_0 \setminus T]$.

For every minimal separator $S \in \mathfrak{T}$ the vertices in $C \cup \bigcup \mathfrak{T} \setminus S$ are in one component of $G - S$ since $\mathfrak{S}$ is a blocking set of G and C is the set of vertices in the interior of $\mathfrak{S}$. Moreover, by Lemma 25 the set $C \cup \bigcup \mathfrak{T} \setminus T$ is in one component of $G - T$. Hence $\{T\} \cup \mathfrak{T}$ is a blocking set of G and C is the set of vertices in the interior of $\{T\} \cup \mathfrak{T}$. Therefore $(\{T\} \cup \mathfrak{T}, C)$ is a block of G. Since $H = G(\mathfrak{S}, C)$ every set $S \in \mathfrak{T}$ is a clique in H. This implies $G(\{T\} \cup \mathfrak{T}, C) = H(\{T\}, D)$. $\square$

Let $(\mathfrak{S}, C)$ be a block of G and let T be a minimal separator of $H = G(\mathfrak{S}, C)$ such that there is a unique minimal separator $S_0 \in \mathfrak{S}$ with $T \subset S_0$. Based on Proposition 29 we define

$$\mathsf{Dec}(\mathfrak{S}, C, T) = \{(\{S_0\}, \varnothing), (\{T\} \cup \mathfrak{S} \setminus \{S_0\}, C)\}.$$

Proposition 30 (Type 3). *Let $(\mathfrak{S}, C)$ be a block of G and let T be a minimal separator of $H = G(\mathfrak{S}, C)$ such that $T \setminus S \neq \varnothing$ for all $S \in \mathfrak{S}$. Let $H[D]$ be a component of $H - T$. Let $\mathfrak{T} = \{S : S \in \mathfrak{S} \text{ and } S \cap D \neq \varnothing\}$ and $D' = D \setminus \bigcup \mathfrak{T}$. Then $(\{T\} \cup \mathfrak{T}, D')$ is a block of G and $G(\{T\} \cup \mathfrak{T}, D') = H(\{T\}, D)$.*

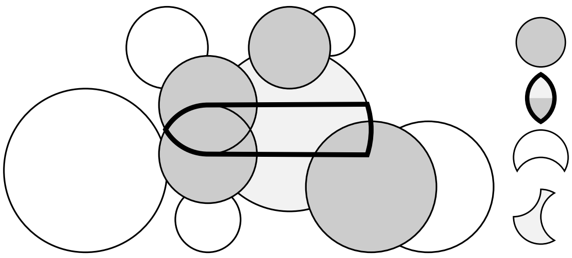

Figure 3: Type 3

Proof. First we show that $\{T\} \cup \mathfrak{T}$ is a blocking set of G. By Lemma 26, T is a minimal separator of G, and every element of $\mathfrak{T}$ is a minimal separator of G. By the presumption of the proposition, the minimal separators in $\{T\} \cup \mathfrak{T}$ are pairwise incomparable. The set $\bigcup \mathfrak{T} \setminus T$ is in one component of $G - T$ by Lemma 25. For every $S \in \mathfrak{T}$ the set $(T \cup \bigcup \mathfrak{T}) \setminus S$ is in one component of $G - S$ since $\mathfrak{S}$ is a blocking set of G and $T \setminus \bigcup \mathfrak{S} \subseteq C$. Hence $\{T\} \cup \mathfrak{T}$ is a blocking set of G.

If $\mathfrak{T} = \varnothing$ then $D \cap \bigcup \mathfrak{S} = \varnothing$ and $G[D]$ is a component of $G - T$. Consequently, $D' = D$ and $(\{T\}, D) = (\{T\} \cup \mathfrak{T}, D')$ is a block of G with $G(\{T\} \cup \mathfrak{T}, D') = H(\{T\}, D)$.

Otherwise $\mathfrak{T} \neq \varnothing$. Then $\mathfrak{S} \neq \varnothing$ since $\mathfrak{T} \subseteq \mathfrak{S}$. Next we show that a vertex v is in the interior of $\{T\} \cup \mathfrak{T}$ if and only if $v \in D'$. A vertex v in the interior of $\{T\} \cup \mathfrak{T}$ belongs to the component of $G - T$ containing $\bigcup \mathfrak{T} \setminus T$. Furthermore, for every $S \in \mathfrak{T}$ the vertex v belongs to the component of $G - S$ containing $T \setminus S$. Consequently $v \in D'$ since $v \notin T \cup \bigcup \mathfrak{T}$.

Let v be a vertex in D'. First assume $v \in C$. Then there exists a minimal separator $S \in \mathfrak{S}$ such that $v \in S$ since $D \subseteq (C \cup \bigcup \mathfrak{S}) \setminus T$. Then $S \in \mathfrak{T}$ by the definition of $\mathfrak{T}$. But this implies $v \in \bigcup \mathfrak{T}$, contradicting $v \in D'$. Consequently, $v \in C$. Now, for all $S \in \mathfrak{T}$, there is one component $G[B]$ of $G - S$ with $C \subseteq B$ and $\bigcup \mathfrak{S} \setminus S \subseteq B$. Then $(T \cup \bigcup \mathfrak{T}) \setminus S \subseteq B$ since $T \subseteq C \cup \bigcup \mathfrak{S}$. This component $G[B]$ of $G - S$ contains the vertex $v \in D'$ since $v \in C$. Furthermore, $v \in D'$ belongs to the component of $G - T$ containing $\bigcup \mathfrak{T} \setminus T$ by Lemma 25 and $D' \subseteq D$. This implies that every vertex $v \in D'$ is in the interior of $\{T\} \cup \mathfrak{T}$.

Consequently, $(\{T\} \cup \mathfrak{T}, D')$ is a block of G. Since $H = G(\mathfrak{S}, C)$ every set $S \in \mathfrak{T}$ is complete in H. This implies $G(\{T\} \cup \mathfrak{T}, D') = H(\{T\}, D)$. $\square$

Let $(\mathfrak{S}, C)$ be a block of G and let T be a minimal separator of $H = G(\mathfrak{S}, C)$ such that $T \setminus S \neq \varnothing$ for all $S \in \mathfrak{S}$. In this case let $\{H[D_i] : i \in I\}$ be the set of components of $H - T$. Based on Proposition 30 we define

$$\mathsf{Dec}(\mathfrak{S}, C, T) = \{(\{T\} \cup \{S : S \in \mathfrak{S} \text{ and } S \cap D_i \neq \varnothing\}, C \cap D_i) : i \in I\}.$$

The following theorem summarizes Lemma 27 and Propositions 28, 29 and 30.

Theorem 31. *Let $(\mathfrak{S}, C)$ be a block of G and let T be a minimal separator of $H = G(\mathfrak{S}, C)$. Then we have a bijection between the blocks (T, D) corresponding with the components of $H - T$ and the blocks $(\mathfrak{T}, D')$ in $\mathsf{Dec}(\mathfrak{S}, C, T)$ such that $G(\mathfrak{T}, D') = H(\{T\}, D)$.*

6 Algorithms

The approach of the previous section enables two different types of algorithms. One type is a dynamic programming algorithm as used in [8, 12, 18, 19]. Typical for these algorithms is a step sorting the blocks by the number of vertices in their realization. Then it is possible to compute a parameter like the treewidth of the realization by evaluating the recurrence relation and looking up the values for smaller realizations.

Here we use another type of algorithm sometimes called recursive algorithm with memoization. First we describe the generic version. The input is a graph G. In a preprocessing the algorithm computes $\mathsf{Sep}(G)$ using the listing algorithm given in [16].

The procedure `compute` is the heart of the algorithm. It is recursive via `access`. Both `compute` and `main` use the macros `collect`, `complete`, `initialize`, `update` and `start`, which are specific to the algorithmic problem under consideration.

The algorithm uses a data structure X that can store any block $(\mathfrak{S}, C)$ of G with a value $p(\mathfrak{S}, C)$, and retrieve these values. Suppose $V = \{1, 2, \ldots, n\}$. Any block $(\mathfrak{S}, C)$ is stored as a set $C \subseteq V$ followed by a sequence of the minimal separators $S_1, S_2, \ldots, S_j$ in $\mathfrak{S}$ that are lexicographically ordered (as subsets of V). The data structure X supports the following operations:

- `store`$(\mathfrak{S}, C, p)$ stores for the block $(\mathfrak{S}, C)$ the value p,
- `present`$(\mathfrak{S}, C)$ returns **true**, if an operation `store`$(\mathfrak{S}, C, p)$ has been performed before, for any value of p, and **false** otherwise, and

```
procedure main;
begin
  list Sep(G);
  p ← start;
  for C ∈ Comp(G) do p ← collect(access(∅, C));
  return(p)
end.

procedure access(𝔖, C);
begin
  if not present(𝔖, C) then compute(𝔖, C);
  return(value(𝔖, C))
end;

procedure compute(𝔖, C);
begin
  p ← complete;
  if G(𝔖, C) is not complete then
    for T ∈ Sep(G) do
      if T is a minimal separator of G(𝔖, C) then
        begin
          q ← initialize;
          for (𝔗, D) ∈ Dec(𝔖, C, T) do q ← update(access(𝔗, D));
          p ← min{p, q};
        end;
  store(𝔖, C, p)
end;
```

- `value`$(\mathfrak{S}, C)$ returns the value p of the (last) operation `store`$(\mathfrak{S}, C, p)$, if `present`$(\mathfrak{S}, C) =$ **true**.

All three operations can be executed by iterated search for a vertex in the universe V. A single search can be done in time $O(\log n)$ by standard techniques. To find a whole block $(\mathfrak{S}, C)$ we need $|C| + |\bigcup \mathfrak{S}|$ single searches if $|\mathfrak{S}| \leq 1$ and $\sum_{S \in \mathfrak{S}} |S|$ single searches if $|\mathfrak{S}| \geq 2$. We refer to [3] for an implementation of a related data structure that can easily be extended to one satisfying our purposes. Notice that our algorithm calls `value`$(\mathfrak{S}, C)$ only if `present`$(\mathfrak{S}, C) =$ **true**. Furthermore if `store`$(\mathfrak{S}, C)$ is called, then `present`$(\mathfrak{S}, C) =$ **false**, i.e., for each block of G, `store` is called at most

	treewidth	minimum fill-in	ranking number
`collect`(c)	$\max\{p,c\}$	$p+c$	$\max\{p,c\}$
`complete`	$\lvert C \cup \bigcup \mathfrak{S}\rvert - 1$	$\mathsf{fill}(C \cup \bigcup \mathfrak{S})$	$\lvert C\rvert$
`initialize`	0	$\mathsf{fill}(T)$	$\lvert T \cap C\rvert$
`update`(c)	$\max\{q,c\}$	$q+c-\mathsf{fill}(T)$	$\max\{q, c+\lvert T\cap C\rvert\}$
`start`	0	0	0

once.

We consider the running time of our algorithm on an input graph $G = (V, E)$ with $|V| = n$, $|E| = m$, $|\mathsf{Sep}(G)| = r$ and $\mathsf{an}(G) = k$. First the algorithm in [16] needs $O(n^5 r + m)$ time to list all minimal separators of G.

For the following analysis, we assume that all macros can be evaluated in constant time. (If this is not the case in a particular application, it should be easy to achieve the corresponding time bound with a similar analysis.) To determine the overall running time, we estimate the running time of `compute`$(\mathfrak{S}, C)$ for any block $(\mathfrak{S}, C)$ of G without counting the running time of those recursive calls `compute`$(\mathfrak{T}, D)$ for which `present`$(\mathfrak{T}, D) =$ **false** when `compute`$(\mathfrak{T}, D)$ is called. For any block $(\mathfrak{S}, C)$ of G, `access` calls `compute` at most once, namely when `present`$(\mathfrak{S}, C) =$ **false**. In this case, for each minimal separator T of G, `compute` needs $O(n + m)$ time to test whether T is a minimal separator of $G(\mathfrak{S}, C)$ and, if so, to compute the blocks in $\mathsf{Dec}(\mathfrak{S}, C, T)$. For each of the at most n blocks $(\mathfrak{T}, D)$ in $\mathsf{Dec}(\mathfrak{S}, C, T)$, `access`$(\mathfrak{T}, D)$ is executed. If `access` is called for a block $(\mathfrak{T}, D)$ of G, when `present`$(\mathfrak{T}, D) =$ **true**, then `access` does not call `compute`.

Procedure `access` looks up the value $p(\mathfrak{S}, C)$ in the data structure X. Using an implementation of the data structure X, similar to the one described in [3], one look-up can be done in time $\sum_{S \in \mathfrak{S}} |S| \cdot O(\log n) = O(kn \log n)$.

By Observation 22, the number of different blocks of the input graph G is at most $(r + 1)n + \sum_{i=2}^{k} \binom{r}{i}$. Consequently, the total running time of the algorithm is $O(n^5 r + m + kr^{k+1}(n + m)n \log n)$.

Theorem 32. *On an input graph $G = (V, E)$ with r minimal separators the generic algorithm runs in time $O(n^5 r + m + kr^{k+1}(n + m)n \log n)$ (under some assumptions on the macros), where $|V| = n$, $|E| = m$, and $\mathsf{an}(G) = k$.*

The generic algorithm can be used to compute a graph parameter which can be evaluated via a certain type of recurrence involving the minimal separators of the graph (see e.g. Section 3).

6.1 Treewidth and Minimum fill-in

The problems Treewidth and Minimum fill-in are typical problems that can be solved by the generic algorithm, since the recurrences in Theorems 15 and 16 indeed require to add edges such that the minimal separator becomes a clique. This corresponds exactly to the execution of the generic algorithm.

First we consider the problem Treewidth. For every graph G we have

$$\mathsf{tw}(G) = \max_{C \in \mathsf{Comp}(G)} \mathsf{tw}(G[C]).$$

Since for all components $G[C]$ of G we have $G[C] = G(\varnothing, C)$ this is correctly computed by our choice of `collect`. By Theorems 15 and 31, the generic algorithm correctly computes the treewidth of the input graph. If $G(\mathfrak{S}, C)$ is complete, then $\mathsf{tw}(G(\mathfrak{S}, C)) = |C \cup \bigcup \mathfrak{S}| - 1$, as evaluated by `complete`; otherwise

$$\mathsf{tw}(G(\mathfrak{S}, C)) = \min_{T \in \mathsf{Sep}(G(\mathfrak{S},C))} \max_{(\mathfrak{T},D) \in \mathsf{Dec}(\mathfrak{S},C,T)} \mathsf{tw}(G(\mathfrak{T}, D)).$$

This is evaluated by `initialize` and `update`.

Now we consider the problem Minimum fill-in. It is easy to see that

$$\mathsf{mfi}(G) = \sum_{C \in \mathsf{Comp}(G)} \mathsf{mfi}(G[C]).$$

This is computed by `collect`. By Theorems 16 and 31, the generic algorithm correctly computes the minimum fill-in of the input graph. If $G(\mathfrak{S}, C)$ is complete, then $\mathsf{mfi}(G(\mathfrak{S}, C)) = \mathsf{fill}(C \cup \bigcup \mathfrak{S})$, as evaluated by `complete`; otherwise

$$\mathsf{mfi}(G(\mathfrak{S}, C)) = \min_{T \in \mathsf{Sep}(G(\mathfrak{S},C))} \Big(\mathsf{fill}(T) + \sum_{(\mathfrak{T},D) \in \mathsf{Dec}(\mathfrak{S},C,T)} \big(\mathsf{mfi}(G(\mathfrak{T}, D)) - \mathsf{fill}(T)\big)\Big),$$

which is correctly evaluated by the choice of `initialize` and `update`.

6.2 Vertex ranking

Our first goal in designing the generic algorithm was to be able to handle minimal separators of blocks $G(\mathfrak{S}, C)$ as they occur in Theorems 15 and 16. In contrast, in Theorem 17, minimal separators of components $G[C]$ are needed. Fortunately it turns out that the generic algorithm can also be applied to some problems with recurrences on minimal separators as the one for Vertex ranking.

For the correctness proof we need the following proposition.

Proposition 33. *Let $(\mathfrak{S}, C)$ be a block of G and let R be a minimal separator of $G[C]$. Then there is a minimal separator T of $G(\mathfrak{S}, C)$ such that*

$$\mathsf{Comp}(G[C] - R) = \bigcup_{(\mathfrak{T},D)\in\mathsf{Dec}(\mathfrak{S},C,T)} \mathsf{Comp}(G[D]).$$

Proof. Let R be a minimal a, b-separator of $G[C]$. Then $R \cup \bigcup\mathfrak{S}$ is an a, b-separator of $H = G(\mathfrak{S}, C)$. We choose a minimal a, b-separator T of H with $T \subseteq R \cup \bigcup\mathfrak{S}$. We claim $R \subseteq T$. We consider a vertex $t \in R \setminus T$. Then $G[C] - (R \setminus \{t\})$ contains a path P from a to b via t since R is a minimal a, b-separator of $G[C]$. But P is also a path from a to b in $H - T$, contradicting the fact that T is a minimal a, b-separator of H. Hence a vertex $t \in R \setminus T$ cannot exist. This implies $R \subseteq T$, more precisely, $T \cap C = R$ and $T \setminus C \subseteq \bigcup\mathfrak{S}$.

Now let $G[B]$ be a component of $G[C] - R$. Then there is a component $H[D']$ of $H - T$ with $B \subseteq D'$ since $T \setminus C \subseteq \bigcup\mathfrak{S}$. Furthermore $B \subseteq D' \setminus \bigcup\mathfrak{S}$ since $B \subset C$.

By Theorem 31 there is a block $(\mathfrak{T}, D)$ of G in the set $\mathsf{Dec}(\mathfrak{S}, C, T)$ with $G(\mathfrak{T}, D) = H(\{T\}, D')$. $(\mathfrak{T}, D) \in \mathsf{Dec}(\mathfrak{S}, C, T)$ implies $\mathfrak{T} \subseteq \{T\} \cup \mathfrak{S}$ and $D \cap (T \cup \bigcup\mathfrak{S}) = \varnothing$. This ensures $B \subseteq D$. It remains to show that $G[B]$ is a component of $G[D]$.

Since $G[B]$ is connected it is contained in a component of $G[D]$. We consider an edge $\{b, d\}$ of G with $b \in B$ and $d \in D$. Note that $N(B) \setminus B \subseteq R \cup \bigcup\mathfrak{S}$. The vertex d cannot belong to R or $\bigcup\mathfrak{S}$ since $R \subseteq T$ and $D \cap (T \cup \bigcup\mathfrak{S}) = \varnothing$. Hence $d \in B$ and $G[B]$ is a component of $G[D]$.

Finally let $G[B]$ be a component of $G[D]$ for a block $(\mathfrak{T}, D) \in \mathsf{Dec}(\mathfrak{S}, C, T)$. Then $B \subseteq C \setminus R$ since $D \subseteq C \setminus T$. The graph $G[B]$ is connected, hence B is contained in one component of $G[C] - R$. We consider an edge $\{b, c\}$ of G with $b \in B$ and $c \in C \setminus R$. Note that $N(B) \setminus B \subseteq \bigcup\mathfrak{T}$ since $G[B]$ is a component of $G[D]$. The vertex c cannot belong to $\bigcup\mathfrak{T}$ since $\mathfrak{T} \subseteq \{T\} \cup \mathfrak{S}$, $C \cap \bigcup\mathfrak{S} = \varnothing$ and $c \in C \setminus T$. Hence $c \in B$ and $G[B]$ is a component of $G[C] - R$. □

Recalling Lemma 26 and Theorems 17 and 31 we obtain the following recurrence.

Corollary 34. *For every block $(\mathfrak{S}, C)$ of G we have*

$$\chi_{\mathrm{r}}(G[C]) = \min_{T\in\mathsf{Sep}(G(\mathfrak{S},C))} \big(|T \cap C| + \max_{(\mathfrak{T},D)\in\mathsf{Dec}(\mathfrak{S},C,T)} \chi_{\mathrm{r}}(G[D])\big).$$

Clearly $\chi_{\mathrm{r}}(G) = \max_{C\in\mathsf{Comp}(G)} \chi_{\mathrm{r}}(G[C])$. This is evaluated by `collect`. If $G[C]$ is complete, then $\chi_{\mathrm{r}}(G[C]) = |C|$, as computed by `complete`; otherwise `initialize` and `update` evaluate the above recurrence.

The following theorem summarizes the main results of this section.

Theorem 35. *For each of the problems* Treewidth, Minimum fill-in *and* Vertex ranking *there is an algorithm to compute the corresponding graph parameter for any input graph* G *in time* $O(n^5 r + m + kr^{k+1}(n+m)n \log n)$*, where* r *is the number of minimal separators of* G *and* $k = \mathsf{an}(G)$*.*

References

[1] Arnborg, S., Efficient algorithms for combinatorial problems on graphs with bounded decomposability—A survey. *BIT* **25** (1985), pp.2–23.

[2] Arnborg, S., D. G. Corneil and A. Proskurowski, Complexity of finding embeddings in a k-tree, *SIAM Journal on Algebraic and Discrete Methods* **8** (1987), pp. 277–284.

[3] Bodlaender, H., Kayles on special classes of graphs - An application of Sprague-Grundy theory. *Proceedings of the 18th International Workshop on Graph-Theoretic Concepts in Computer Science, WG'92*, Springer-Verlag, LNCS 657 (1993), pp. 90–102.

[4] Bodlaender, H., A tourist guide through treewidth, *Acta Cybernetica* **11** (1993), pp. 1–23.

[5] Bodlaender, H., A linear time algorithm for finding treedecompositions of small treewidth, *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, (1993), pp. 226–234.

[6] Bodlaender, H., J. S. Deogun, K. Jansen, T. Kloks, D. Kratsch, H. Müller, Zs. Tuza, Rankings of graphs, *Proceedings of the 20th International Workshop on Graph-Theoretic Concepts in Computer Science, WG'94*, Springer-Verlag, LNCS 903 (1995), pp. 292–304, full version to appear in *SIAM Journal on Discrete Mathematics* 1998.

[7] Bodlaender, H. L., J. R. Gilbert, H. Hafsteinsson, T. Kloks, Approximating treewidth, pathwidth and minimum elimination tree height, *Journal of Algorithms* **18** (1995), pp. 238–255.

[8] Bodlaender, H., T. Kloks and D. Kratsch, Treewidth and pathwidth of permutation graphs, *SIAM Journal on Discrete Mathematics* **8** (1995), pp. 606–616.

[9] Broersma, H. J., T. Kloks, D. Kratsch and H. Müller, Independent sets in asteroidal triple-free graphs, *Proceedings of the 24th International Colloquium on Automata, Languages, and Programming (ICALP'97)*, Springer-Verlag, LNCS 1256 (1997), pp. 760–770.

[10] Corneil, D. G., S. Olariu and L. Stewart, Asteroidal triple-free graphs, *SIAM Journal on Discrete Mathematics* **10** (1997), pp. 399–430.

[11] Corneil, D. G., S. Olariu and L. Stewart, A linear time algorithm to compute dominating pairs in asteroidal triple-free graphs, *Proceedings of ICALP'95*, Springer-Verlag, LNCS 944 (1995), pp. 292–302.

[12] Deogun, J. S., T. Kloks, D. Kratsch and H. Müller, On vertex ranking for permutation and other graphs, *11th Annual Symposium on Theoretical Aspects of Computer Science*, Springer-Verlag, LNCS 775 (1994), pp. 747–758.

[13] Dirac, G. A., On rigid circuit graphs, *Abhandlungen des Mathematischen Sememinars der Universität Hamburg* **25** (1961), pp. 71–76.

[14] Golumbic, M. C., *Algorithmic graph theory and perfect graphs*, Academic Press, New York, 1980.

[15] Kloks, T., *Treewidth – Computations and Approximations*, Springer-Verlag, LNCS 842 (1994).

[16] Kloks, T. and D. Kratsch, Finding all minimal separators of a graph, *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science, STACS'94*, Springer-Verlag, LNCS 775 (1994), pp. 759–768, full version to appear in *SIAM Journal on Computing* 1998.

[17] Kloks, T., D. Kratsch and H. Müller, Asteroidal sets in graphs, *Proceedings of the 19th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'97)*, Springer-Verlag, LNCS 1335 (1997), pp. 229–241.

[18] Kloks, T., D. Kratsch and J. Spinrad, On treewidth and minimum fill-in of asteroidal triple-free graphs, *Theoretical Computer Science* **175** (1997), pp. 309-335.

[19] Kloks, T., H. Müller and C. K. Wong, Vertex ranking of asteroidal triple-free graphs, *Proceedings of the 7th International Symposium on Algorithms and Computation, ISAAC'96*, Springer-Verlag, LNCS 1178 (1996), pp. 174–182.

[20] Leiserson, C. E., Area efficient graph layouts for VLSI, *Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science*, 1980, pp. 270–281.

[21] Lekkerkerker, C. G. and J. Ch. Boland, Representation of a finite graph by a set of intervals on the real line, *Fundamenta Mathematicae* **51** (1962), pp. 45–64.

[22] Lin, I. J., T. A. McKee and D. B. West, Leafage of chordal graphs, Manuscript 1994.

[23] Liu, J. W. H., The role of elimination trees in sparse factorization, *SIAM Journal of Matrix Analysis and Applications* **11** (1990), pp. 134–172.

[24] Möhring, R. H., Triangulating graphs without asteroidal triples, *Discrete Applied Mathematics* **64** (1996), pp. 281–287.

[25] Prisner, E., Representing triangulated graphs in stars, *Abhandlungen des Mathematischen Seminars der Universität Hamburg* **62** (1992), pp. 29–41.

[26] Rose, D. J., R. E. Tarjan and G. S. Luecker, Algorithmic aspects of vertex elimination on graphs, *SIAM Journal on Computing* **5**, (1976), pp. 266–283.

[27] Walter, J. R., Representations of chordal graphs as subtrees of a tree, *Journal of Graph Theory* **2** (1978), pp. 265–267.

[28] Yannakakis, M., Computing the minimum fill-in is NP-complete, *SIAM Journal on Algebraic and Discrete Methods* **2** (1981), pp. 77–79.