Paper V

Exact algorithms for treewidth and minimum fill-in*†

Fedor V. Fomin‡
Department of Informatics
University of Bergen
5020 Bergen, Norway
fomin@ii.uib.no

Dieter Kratsch
LITA, Université de Metz
57045 Metz Cedex 01, France
kratsch@univ-metz.fr

Ioan Todinca
LIFO, Université d'Orléans
45067 Orléans Cedex 2, France
Ioan.Todinca@lifo.univ-orleans.fr

Yngve Villanger
Department of Informatics
University of Bergen
5020 Bergen, Norway
yngvev@ii.uib.no

Abstract

We show that there are $\mathcal{O}(1.8899^n)$ time algorithms to compute the treewidth and the minimum fill-in of each graph G on n vertices. Our result is based on a combinatorial proof that each graph on n vertices has at most $n \cdot 1.7087^n$ minimal separators and that all potential maximal cliques can be listed in $\mathcal{O}(1.8899^n)$ time. For the class of AT-free graphs we obtain $\mathcal{O}(1.4142^n)$ time algorithms to compute treewidth and minimum fill-in.

Keywords: Exact exponential algorithm, treewidth, fill-in, minimal separators, potential maximal clique, minimal triangulation

*A preliminary version [26] of this paper has been presented at the 31st International Colloquium on Automata, Languages and Programming, Turku, Finland, July 2004.

†Supported by The Aurora Programme Collaboration Research Project between Norway and France.

‡Fedor Fomin acknowledges support of Norges forskningsråd, project 160778/V30.

1 Introduction

Exact exponential algorithms. The interest in exact (fast) exponential algorithms dates back to Held and Karp's paper [29] on the travelling salesman problem in the early sixties. Mention just a few examples: time $\mathcal{O}^*(1.4422^n)$ algorithm for Knapsack (Horowitz and Sahni [30]); time $\mathcal{O}^*(1.2600^n)$ and $\mathcal{O}^*(1.2109^n)$ algorithms for Independent Set (Tarjan and Trojanowski [44], Robson [40]); 3-Coloring in time $\mathcal{O}^*(1.4422^n)$ (Lawler [34]); 3-SAT in time $\mathcal{O}^*(1.6181^n)$ (Monien and Speckenmeyer [35]).

In this paper we use a modified big-Oh notation that suppresses all other (polynomially bounded) terms. For functions f and g we write $f(n) = \mathcal{O}^*(g(n))$ if $f(n) = \mathcal{O}(g(n)poly(n))$, where $poly(n)$ is a polynomial. This modification may be justified by the exponential growth of $f(n)$.

Nowadays, it is common believe that NP-hard problems can not be solved in polynomial time. For a number of NP-hard problems, we even have strong evidence that they cannot be solved in sub-exponential time. In order to obtain exact solutions to these problems, the only hope is to design exact algorithms with good exponential running times. How good can these exponential running times be? Can we reach 2^{n^2} for instances of size n? Can we reach 10^n? Or even 2^n? Or can we reach c^n for some constant c that is very close to 1? The last years have seen an emerging interest in attacking these questions for concrete combinatorial problems: There is an $\mathcal{O}^*(2.4150^n)$ time algorithm for Coloring (Byskov [15]); an $\mathcal{O}^*(1.3289^n)$ time algorithm for 3-Coloring (Beigel and Eppstein [3]); an $\mathcal{O}^*(1.7325^n)$ time algorithm for Max-Cut (Williams [47]); an algorithms for 3-SAT in time $\mathcal{O}^*(1.4726^n)$ (Brueggemann and Kern [14]); an $\mathcal{O}^*(1.5129^n)$ time algorithm for Dominating Set (Fomin et al. [25]).

There can be several explanations why now the algorithmic community witnesses the revival of the interest in fast exponential algorithms:

- The design and analysis of exact algorithms leads to a better understanding of NP-hard problems and initiates interesting new combinatorial and algorithmic challenges.
- For certain applications it is important to find exact solutions. With the increased speed of modern computers, fast algorithms, even though they have exponential running times in the worst case, may actually lead to practical algorithms for certain NP-hard problems, at least for moderate instance sizes.
- Approximation, randomized algorithms and different heuristics are not always satisfactory. Each of these approaches has weak points like necessity of exact solutions, difficulty of approximation, limited power of the method itself and many others.

- A reduction of the base of the exponential running time, say from $\mathcal{O}(2^n)$ to $\mathcal{O}(1.8^n)$, increases the size of the instances solvable within a given amount of time by a constant *multiplicative* factor. However running a given exponential algorithm on a faster computer can enlarge the mentioned size only by a constant *additive* factor.

For overviews and introductions to the field see the recent surveys by Iwama [31], Schöning [42], and Woeginger [48, 49].

Treewidth and minimum fill-in. Treewidth is one of the most basic parameters in Graph Algorithms [6] and it plays an important role in structural Graph Theory. It serves as one of the main tools in Robertson and Seymour's Graph Minors project [39]. Treewidth also plays a crucial role in parameterized complexity theory [21]. The minimum fill-in problem (also known as minimum chordal graph completion) has important applications in sparse matrix computations and computational biology.

The problems to compute the treewidth and minimum fill-in of a graph are known to be NP-hard even when the input is restricted to complements of bipartite graphs (so called cobipartite graphs) [2, 50]. Despite of the importance of treewidth almost nothing is known on how to cope with its intractability. For a long time the best known approximation algorithm for treewidth had a factor $\log OPT$ [1, 11] (see also [7]). Recently, Feige et al. [23] obtained factor $\sqrt{\log OPT}$ approximation algorithm for treewidth. Furthermore it is an old open question whether the treewidth can be approximated within a constant factor.

Treewidth is known to be fixed parameter tractable. Moreover, for any fixed k, there is a linear time algorithm to compute the treewidth of graphs of treewidth at most k (unfortunately there is a huge hidden constant in the running time) [5]. There is a number of algorithms that for a given graph G and integer k, either report that the treewidth of G is at least k, or produce a tree decomposition of width at most $c_1 \cdot k$ in time $c_2{}^k \cdot n^{O(1)}$, where c_1, c_2 are some constants (see e.g. [1]). Fixed parameter algorithms are known for the minimum fill-in problem as well [17, 32].

We are not aware about any previous work on exact algorithms for the treewidth or minimum fill-in problem. There are three relatively simple approaches resulting in time $\mathcal{O}^*(2^n)$ algorithms:

- One can reformulate the problems as problems of finding special vertex elimination orderings and then find an optimal permutation by using the dynamic programming based technique like in the article of Held & Karp [29] for the travelling salesman problem;
- With some modifications, the algorithm of Arnborg et al. [2] for a given k deciding in time $O(n^{k+1})$ if the treewidth of a graph is at most k, can be used to compute the treewidth (and similarly fill-in) in time $\mathcal{O}^*(2^n)$;

- Both problems can be solved by making use of game theoretic approach, by finding a specific path in the graph of possible states of Cop and Robber game [24].

However it is not clear if any of the mentioned approaches can bring us to time $\mathcal{O}^*(c^n)$ algorithm for some $c < 2$.

Our results. In this paper we obtain the first exact algorithm computing the treewidth in time $\mathcal{O}^*(c^n)$ for $c < 2$. Additionally it can be adapted to solve a number of other minimal triangulation problems like minimum fill-in.

Our main result is an $\mathcal{O}^*(1.8899^n)$ algorithm computing the treewidth and minimum fill-in of a graph on n vertices. The algorithm can be regarded as dynamic programming across partial solutions and is based on results of Bouchitté & Todinca [9, 10]. The analysis of the running time is difficult and is based on combinatorial properties of special structures in a graph, namely, potential maximal clique, i.e. vertex subsets in a graph that can be maximal cliques in some minimal triangulation of this graph. (See the next section for the definition.)

More precisely, first we modify the algorithm of Bouchitté & Todinca [9] which computes the treewidth and minimum fill-in of a graph G with the given set Π_G of all potential maximal cliques of G and then improve the analysis of its running time to obtain an $\mathcal{O}^*(|\Pi_G|)$ time complexity. The most technical and difficult part of the paper is the proof that all potential maximal cliques can be listed in time $\mathcal{O}^*(1.8899^n)$. Very roughly, our listing algorithms is based on the following combinatorial result: every "large" potential maximal clique is either "almost" a minimal separator, or can be represented by a "small" vertex subset. The technique developed to prove this combinatorial result can be interesting on its own.

For several special graph classes, for which both problems remain NP-complete, we are able to prove that our approach guarantees significantly better bounds. To exemplify this we show that for the class of AT-free graphs the number of minimal separators and the number of potential maximal cliques, and thus the running time of our algorithm, is $\mathcal{O}^*(2^{n/2})$.

This paper is organized as follows. In Section 2 we give basic definitions. In Section 3 we show how Bouchitté & Todinca's approach can be used to compute the treewidth and fill-in in time linear in the number of potential maximal cliques. In Section 4 we prove that every graph on n vertices has $\mathcal{O}(n \cdot 1.7087^n)$ minimal separators. The results of Section 4 are used in Section 5, where we derive the most difficult and important combinatorial result of this paper, namely, that all potential maximal cliques of a graph can be listed in time $\mathcal{O}^*(1.8899^n)$. Combining with the results from Section 3, this yields the main result of the paper, that the treewidth and minimum fill-in can be computed in time $\mathcal{O}^*(1.8899^n)$. In

Section 6 we design faster $\mathcal{O}^*(2^{n/2})$ algorithm on AT-free graphs. We conclude with open problems and final remarks in Section 7.

2 Basic definitions

We denote by $G = (V, E)$ a finite, undirected and simple graph with $|V| = n$ vertices and $|E| = m$ edges. For any non-empty subset $W \subseteq V$, the subgraph of G induced by W is denoted by $G[W]$. For $S \subseteq V$ we often use $G \setminus S$ to denote $G[V \setminus S]$. The *neighborhood* of a vertex v is $N(v) = \{u \in V : \{u, v\} \in E\}$ and for a vertex set $S \subseteq V$ we set $N(S) = \bigcup_{v \in S} N(v) \setminus S$. A *clique* C of a graph G is a subset of V such that all the vertices of C are pairwise adjacent. By $\omega(G)$ we denote the maximum clique-size of a graph G.

Treewidth and minimum fill-in of graphs. The notion of treewidth is due to Robertson & Seymour [38]. A *tree decomposition* of a graph $G = (V, E)$, denoted by $TD(G)$, is a pair (X, T) in which $T = (V_T, E_T)$ is a tree and $X = \{X_i | i \in V_T\}$ is a family of subsets of V such that:

(i) $\bigcup_{i \in V_T} X_i = V$;

(ii) for each edge $e = \{u, v\} \in E$ there exists an $i \in V_T$ such that both u and v belong to X_i; and

(iii) for all $v \in V$, the set of nodes $\{i \in V_T | v \in X_i\}$ induces a connected subtree of T.

The maximum of $|X_i| - 1$, $i \in V_T$, is called the *width* of the tree decomposition. The *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, is the minimum width taken over all tree decompositions of G.

A graph H is *chordal* (or *triangulated*) if every cycle of length at least four has a chord, i.e. an edge between two non-consecutive vertices of the cycle. A *triangulation* of a graph $G = (V, E)$ is a chordal graph $H = (V, E')$ such that $E \subseteq E'$. H is a *minimal triangulation* if for any intermediate set E'' with $E \subseteq E'' \subset E'$, the graph $F = (V, E'')$ is not chordal.

The following result is very useful for our algorithms.

Theorem 1 (Folklore). *For any graph G, $\mathrm{tw}(G) \le k$ if and only if there is a triangulation H of G such that $\omega(H) \le k + 1$.*

Thus the treewidth of a graph G can be defined as the minimum of $\omega(H) - 1$ taken over all triangulations H of G, of $\omega(H) - 1$.

The *minimum fill-in* of a graph $G = (V, E)$, denoted by $\mathrm{mfi}(G)$, is the smallest value of $|E_H - E|$, where the minimum is taken over all triangulations $H = (V, E_H)$ of G.

In other words, computing the treewidth of G means finding a (minimal) triangulation with the smallest maximum clique-size, while computing the minimum fill-in means finding a (minimal) triangulation with the smallest number of edges. Clearly, in both cases it is sufficient to consider only minimal triangulations of G, which makes minimal separators and potential maximal cliques important tools of our algorithmic approach.

Minimal separators. Minimal separators and potential maximal cliques are the most important tools used in our proofs. Let a and b be two non adjacent vertices of a graph $G = (V, E)$. A set of vertices $S \subseteq V$ is an *a, b-separator* if a and b are in different connected components of the graph $G \setminus S$. A connected component C of $G \setminus S$ is a *full* component (associated to S) if $N(C) = S$. S is a *minimal a, b-separator* of G if no proper subset of S is an a, b-separator. We say that S is a *minimal separator* of G if there are two vertices a and b such that S is a minimal a, b-separator. Notice that a minimal separator can be strictly included in another one. We denote by Δ_G the set of all minimal separators of G. A set of vertices $\Omega \subseteq V$ of a graph G is called a *potential maximal clique* if there is a minimal triangulation H of G such that Ω is a maximal clique of H. We denote by Π_G the set of all potential maximal cliques of G. Clearly, $|\Delta_G| \leq 2^n$ and $|\Pi_G| \leq 2^n$ for every graph G on n vertices, and no better upper bounds had been known prior to our work.

The following result will be used to list all minimal separators of a graph.

Theorem 2 ([4]). *There is an algorithm listing all minimal separators of an input graph G in $\mathcal{O}(n^3|\Delta_G|)$ time.*

There is a very useful relationship between the minimal separators of a graph and its minimal triangulations. Two minimal separators S and T of a graph G are said to be *crossing* if S is a minimal u, v-separator for a pair of vertices $u, v \in T$, in which case T is a minimal x, y-separator for a pair $x, y \in S$.

Theorem 3 ([37]). *The graph H is a minimal triangulation of the graph G if and only if there is a maximal set of pairwise non-crossing minimal separators $\{S_1, S_2, \ldots, S_p\}$ of G such that H can be obtained from G by completing each S_i, $i \in \{1, 2, \ldots, p\}$, into a clique.*

Although we do not use this characterization explicitly it is fundamental for our paper.

Potential maximal cliques. The following structural characterization of potential maximal cliques is extremely useful for our purposes.

For a set $K \subseteq V$, a connected component C of $G \setminus K$ is a *full component associated to K* if $N(C) = K$.

Theorem 4 ([9]). *Let $K \subseteq V$ be a set of vertices of the graph $G = (V, E)$. Let $\mathcal{C}(K) = \{C_1(K), \dots, C_p(K)\}$ be the set of the connected components of $G \setminus K$ and let $\mathcal{S}(K) = \{S_1(K), S_2(K), \dots, S_p(K)\}$ where $S_i(K)$, $i \in \{1, 2, \dots, p\}$, is the set of those vertices of K which are adjacent to at least one vertex of the component $C_i(K)$. Then K is a potential maximal clique of G if and only if :*

1. *$G \setminus K$ has no full component associated to K, and*
2. *the graph on the vertex set K obtained from $G[K]$ by completing each $S_i \in \mathcal{S}(K)$ into a clique, is a complete graph.*

Moreover, if K is a potential maximal clique, then $\mathcal{S}(K)$ is the set of the minimal separators of G contained in K.

Remark. By Theorem 4, for every potential maximal clique Ω of G, the sets $S_i(\Omega)$ are exactly the minimal separators of G contained in Ω. Let us point out that for each minimal separator $S_i = S_i(\Omega)$, all vertices of $\Omega \setminus S_i$ are contained in the same component of $G \setminus S_i$.

The following result is an easy consequence of Theorem 4.

Theorem 5 ([9]). *There is an algorithm that, given a graph $G = (V, E)$ and a set of vertices $K \subseteq V$, verifies if K is a potential maximal clique of G. The time complexity of the algorithm is $\mathcal{O}(nm)$.*

According to [10], the number of potential maximal cliques of a graph G is at least $|\Delta_G|/n$ and at most $n|\Delta_G|^2 + n|\Delta_G| + 1$. We will show later that a graph on n vertices has $\mathcal{O}^*(1.7087^n)$ minimal separators and $\mathcal{O}^*(1.8899^n)$ potential maximal cliques.

Let us emphasize that it is an open question whether there is an algorithm listing all potential maximal cliques of any graph with a running time $\mathcal{O}(\text{poly}(n) \cdot |\Pi_G|)$ for some polynomial poly(n).

3 Computing treewidth and minimum fill-in

We describe a modification of the algorithm of [9] that, given a graph, all its minimal separators and all its potential maximal cliques, computes the treewidth and the minimum fill-in of the graph. The running time stated in [9] could be reformulated as $\mathcal{O}(n^2\, |\Delta_G| \cdot |\Pi_G|)$. We show how the algorithm can be implemented to run in time $\mathcal{O}(n^3 \cdot |\Pi_G|)$.

For a minimal separator S and a component $C \in \mathcal{C}(S)$ of $G \setminus S$, we say that (S, C) is a *block* associated to S. We sometimes use the notation (S, C) to denote the set of vertices $S \cup C$ of the block. It is easy to notice that if $X \subseteq V$

corresponds the set of vertices of a block, then this block (S, C) is unique: indeed $S = N(V \setminus X)$ and $C = X \setminus S$.

A block (S, C) is called *full* if C is a full component associated to S. The graph $R(S, C) = G_S[S \cup C]$ obtained from $G[S \cup C]$ by completing S into a clique is called the *realization* of the block B.

Theorem 6 ([33]). *Let G be a non-complete graph. Then*

$$\mathrm{tw}(G) = \min_{S \in \Delta_G} \max_{C \in \mathcal{C}(S)} \mathrm{tw}(R(S, C))$$

$$\mathrm{mfi}(G) = \min_{S \in \Delta_G} (\mathrm{fill}(S) + \sum_{C \in \mathcal{C}(S)} \mathrm{mfi}(R(S, C)))$$

where $\mathrm{fill}(S)$ *is the number of non-edges of* $G[S]$.

Remark. In the equations of Theorem 6 we may take the minimum only over the inclusion-minimal separators of G. Then all the blocks in the equations are full.

Unfortunately, Theorem 6 is not sufficient for computing the treewidth and the minimum fill-in. Therefore we now express the treewidth and the minimum fill-in of realizations of full blocks from realizations of smaller full blocks. Let Ω be a potential maximal clique of G. We say that a block (S', C') is *associated to* Ω if C' is a component of $G \setminus \Omega$ and $S' = N(C')$.

Theorem 7 ([9]). *Let (S, C) be a full block of G. Then*

$$\mathrm{tw}(R(S, C)) = \min_{S \subset \Omega \subseteq (S, C)} \max(|\Omega| - 1, \mathrm{tw}(R(S_i, C_i)))$$

$$\mathrm{mfi}(R(S, C)) = \min_{S \subset \Omega \subseteq (S, C)} \Big(\mathrm{fill}(\Omega) - \mathrm{fill}(S) + \sum \mathrm{mfi}(R(S_i, C_i))\Big)$$

where the minimum is taken over all potential maximal cliques Ω such that $S \subset \Omega \subseteq (S, C)$ and (S_i, C_i) are the blocks associated to Ω in G such that $S_i \cup C_i \subset S \cup C$.

Theorem 8. *There is an algorithm that, given a graph G together with the list of its minimal separators Δ_G and the list of its potential maximal cliques Π_G, computes the treewidth and the minimum fill-in of G in $\mathcal{O}(n^3 |\Pi_G|)$ time. Moreover, the algorithm constructs optimal triangulations for the treewidth and the minimum fill-in.*

Proof. W.l.o.g. we may assume that the input graph G is connected (otherwise we can run the algorithm for each connected component of G).

The algorithm for computing the treewidth and the minimum fill-in of a graph, using its minimal separators and its potential maximal cliques is given below. It is a slightly different version of the algorithm given in [9].

```
Input: G, all its potential maximal cliques and all its minimal separators
Output: tw(G) and mfi(G)
begin
    compute all the full blocks (S, C) and sort them by the number of vertices
    for each full block (S, C) taken in increasing order
        tw(R(S, C)) := |S ∪ C| − 1 if (S, C) is inclusion-minimal
            and tw(R(S, C)) := ∞ otherwise
        mfi(R(S, C)) := fill(S ∪ C) if (S, C) is inclusion-minimal
            and mfi(R(S, C)) := ∞ otherwise
        for each p.m.c. Ω with S ⊂ Ω ⊆ (S, C)
            compute the blocks (S_i, C_i) associated to Ω s.t. S_i ∪ C_i ⊂ S ∪ C
            tw(R(S, C)) := min(tw(R(S, C)),
                                max_i(|Ω| − 1, tw(R(S_i, C_i))))
            mfi(R(S, C)) := min(mfi(R(S, C)),
                                fill(Ω) − fill(S) + Σ_i (mfi(R(S_i, C_i))))
        end_for
    end_for
    let Δ*_G be the set of inclusion-minimal separators of G
    tw(G) := min_{S∈Δ*_G} max_{C∈C(S)} tw(R(S, C))
    mfi(G) := min_{S∈Δ*_G} (fill(S) + Σ_{C∈C(S)} mfi(R(S, C)))
end
```

For the sake of completeness we shortly discuss the correctness proof. According to Theorem 7, at the end of the outer `for` loop the values of $\mathrm{tw}(R(S,C))$ and $\mathrm{mfi}(R(S,C))$ are correctly computed, for each full block (S,C) of G. Then the treewidth and the minimum fill-in of the graph are computed using Theorem 6 and the fact that in Theorem 6 one can work only with inclusion-minimal separators.

Let us show how the algorithm can be implemented in time $\mathcal{O}(n^3 \cdot |\Pi_G|)$.

To store and manipulate the minimal separators, potential maximal cliques and blocks we use data structures that allow to search, to insert or to check whether an element is inclusion-minimal in $\mathcal{O}(n)$ time.

During a preprocessing step, we realize the following operations.

- Compute the list of all full blocks and, for each minimal separator S, store a pointer towards each full block of type (S,C). These operations are performed as follows. For each minimal separator S, we compute the connected components of $G \setminus S$. For each such component C, if $N(C) = S$ then the block (S,C) is full, so we add it to the list of full blocks and store the pointer from S to (S,C). Note that this procedure will generate all the full blocks, and each of them is encountered exactly once. For a given minimal separator S, there are at most n full blocks associated to it, so at most n pointers to be stored. The insertion of these blocks into the list of full blocks requires $\mathcal{O}(n)$ time for each block. Hence the whole step costs $\mathcal{O}(n^2|\Delta_G|)$ time.

- For each potential maximal clique Ω, store a pointer to each full block associated to it as follows: compute the components C_i of $G \setminus \Omega$ and then $(N(C_i), C_i)$ are precisely the blocks associated to Ω. In particular there are at most n such blocks. This computation can be done in $\mathcal{O}(n^2)$ time for each potential maximal clique, so globally in $\mathcal{O}(n^2|\Pi_G|)$ time.
- Compute all the *good triples* (S, C, Ω), where (S, C) is a full block and Ω is a potential maximal clique such that $S \subset \Omega \subseteq S \cup C$. Moreover, for each good triple we store a pointer from (S, C) to Ω. By Theorem 4, there are at most n minimal separators $S \subset \Omega$ each of them being the neighborhood of a component of $G \setminus \Omega$ and for each such S there is exactly one component $G \setminus S$ intersecting Ω (in particular there are at most $n|\Pi_G|$ good triples). For each component C' of $G \setminus \Omega$ we take $S = N(C')$, find the component C of $G \setminus S$ intersecting Ω and store the pointer from (S, C) to Ω. Thus this computation takes $\mathcal{O}(nm)$ time for each potential maximal clique, so $\mathcal{O}(nm|\Pi_G|)$ globally.

Hence this preprocessing step costs $\mathcal{O}(n^2|\Delta_G| + nm|\Pi_G|)$. Sorting the blocks by their size can be done in $\mathcal{O}(n|\Delta_G|)$ time using a bucket sort.

Observe that the cost of one iteration of the inner `for` loop is $\mathcal{O}(n^2)$, by the fact that there are at most n blocks associated to a potential maximal clique. With the data structures obtained during the preprocessing step, each full block (S, C) keeps a pointer towards each potential maximal clique Ω such that (S, C, Ω) form a good triple. Thus the number of iterations of the two nested loops is exactly the number of good triples, that is at most $n|\Pi_G|$. It follows that the two loops cost $\mathcal{O}(n^3|\Pi_G|)$ time.

After the execution of the loops, computing the set Δ_G^* of inclusion-minimal separators costs $\mathcal{O}(n|\Delta_G|)$ time. Each inclusion-minimal separator S keeps the list of its associated blocks, obtained during the preprocessing step. Computing the maximum required by the two last instructions costs $\mathcal{O}(n)$ time for a given S. This last step costs $\mathcal{O}(n|\Delta_G|)$ time.

Altogether, the algorithm runs in time $\mathcal{O}(n^2 \cdot |\Delta_G| + n^3 \cdot |\Pi_G|)$. It is known [10] that each minimal separator is contained in at least one potential maximal clique. According to Theorem 4, each potential maximal clique contains at most n minimal separators. Therefore $|\Pi_G| \geq |\Delta_G|/n$. We conclude that the algorithm runs in $\mathcal{O}(n^3 \cdot |\Pi_G|)$ time.

The algorithm can be easily transformed in order to output not only the treewidth and the minimum fill-in of the graph, but also optimal triangulations with respect to these parameters. It is sufficient to keep, for each full block, the set of potential maximal cliques realizing the minimum treewidth and fill-in of its realization. At the end of the algorithm, the potential maximal cliques of the chosen blocks will be the maximal cliques of the computed optimal triangulation: optimal tree decomposition or minimum triangulation. □

Using Theorem 8, the only missing ingredient of our treewidth and minimum fill-in algorithms is an algorithm listing all (minimal separators and) potential maximal cliques of a graph in time $\mathcal{O}^*(c^n)$ for some $c < 2$. This requires exponential upper bounds of the type $\mathcal{O}^*(c^n)$ for some $c < 2$ for the number of minimal separators and for the number of potential maximal cliques in a graph on n vertices. In the next two sections we discuss this issue.

4 Upper bounding the number of minimal separators

In this section we show that any graph with n vertices has $\mathcal{O}(n \cdot 1.7087^n)$ minimal separators. For the main algorithm of this paper the upper bound $\mathcal{O}^*(1.8899^n)$ would be sufficient. However, bounding the number of minimal separators in a graph is a nice combinatorial problem and we prefer to give here the best upper bound we were able to find.

Let S be a separator in a graph $G = (V, E)$. For $x \in V \setminus S$, we denote by $C_x(S)$ the component of $G \setminus S$ containing x. The following lemma is an exercise in [28].

Lemma 9 (Folklore). *A set S of vertices of G is a minimal a, b-separator if and only if a and b are in different full components associated to S. In particular, S is a minimal separator if and only if there are at least two distinct full components associated to S.*

Here is the main combinatorial result.

Theorem 10. *For any graph G, $|\Delta_G| = \mathcal{O}(n \cdot 1.7087^n)$.*

Let us note, that by Theorem 2, Theorem 10 yields that all minimal separators of a graph can be listed in time $\mathcal{O}(n^4 \cdot 1.7087^n)$.

Proof. For a constant α, $0 < \alpha < 1$, we distinguish two types of minimal separators: *small* separators, of size at most αn, and *big* separators, of size more than αn. We denote the cardinalities of these sets by $\#\texttt{small sep}$ and $\#\texttt{big sep}$. Notice that $|\Delta_G| = \#\texttt{small sep} + \#\texttt{big sep}$.

4.1 Upper bounding the number of big separators

Let S be a minimal separator. By Lemma 9, there are at least two full components associated to S. Hence at least one of these full components has at most $n(1-\alpha)/2$ vertices. For every $S \in \Delta_G$ we choose one of these full components, and call it the *small* component of S, denoted by $\mathrm{s}(S)$.

By the definition of a full component, $S = N(\mathrm{s}(S))$. In particular, for distinct minimal separators S and T, we have that $\mathrm{s}(S) \neq \mathrm{s}(T)$. Therefore the number #`big sep` of big minimal separators is at most the number of small components and we conclude that #`big sep` does not exceed the number of subsets of V of cardinality at most $n(1-\alpha)/2$, i.e.

$$\#\texttt{big sep} \leq \sum_{i=1}^{\lceil n(1-\alpha)/2 \rceil} \binom{n}{i}$$

By making use of Stirling's formula we deduce that:

$$\#\texttt{big sep} \leq \frac{n(1-\alpha)}{2} \left(\pi n(1-\alpha)(1+\alpha)/2\right)^{-\frac{1}{2}} \left[\left(\frac{1-\alpha}{2}\right)^{-\frac{1-\alpha}{2}} \left(\frac{1+\alpha}{2}\right)^{-\frac{1+\alpha}{2}} \right]^n$$

4.2 Upper bounding the number of small separators

To count small separators we use a different technique. Let S be a minimal separator, let x be a vertex of a full component $C_x(S)$ associated to S with minimum number of vertices and let $X \subset V$ be a vertex subset. We say that (x, X) is a *bad pair* associated to S if $C_x(S) \subseteq X \subseteq V \setminus S$.

Claim 1. *Let $S \neq T$ be two minimal separators and let (x, X) and (y, Y) be two bad pairs associated to S and T respectively. Then $(x, X) \neq (y, Y)$.*

Proof. Since $C_x(S) \subseteq X$ and $X \cap S = \emptyset$, we have that the connected component of $G[X]$ containing x is $C_x(S)$. Similar, the connected component of $G[Y]$ containing y is $C_y(T)$.

Thus if $x = y$ and $X = Y$, then $C_x(S) = C_y(T)$. Since $C_x(S)$ is a full component associated to S in G, we have that $S = N(C_x(S))$ and $T = N(C_y(T))$. Therefore $S = T$, which is a contradiction. □

By Lemma 9, there are at least two full components associated to every small separator S. For a full component $C_x(S)$ associated to S with the minimum number of vertices, $|V \setminus (S \cup C_x(S))| \geq n \cdot (1-\alpha)/2$. For any $Z \subseteq V \setminus (S \cup C_x(S))$, the pair $(x, Z \cup C_x(S))$ is a bad pair associated to S. Therefore there are at least $2^{n\cdot(1-\alpha)/2}$ distinct bad pairs associated to S. Hence by Claim 1, the total number of bad pairs is at least $\#\texttt{small sep} \cdot 2^{n\cdot(1-\alpha)/2}$. On the other hand, the number of bad pairs is at most $n \cdot 2^n$. We conclude that

$$\#\texttt{small sep} \leq n2^{n\cdot(1+\alpha)/2}$$

Finally, choosing $\alpha = 0.5456$, we obtain

$$|\Delta_G| = \#\texttt{small sep} + \#\texttt{big sep} = \mathcal{O}(n \cdot 1.7087^n).$$

□

5 Upper bounding the number of potential maximal cliques

In this section the we prove the main technical result of this paper, namely that there exists an algorithm to list all potential maximal cliques of any graph in time $\mathcal{O}^*(1.8899^n)$.

Roughly speaking, the idea is to show that each potential maximal clique of a graph can be identified by a set of vertices of size at most $n/3$. The algorithm for generating all the potential maximal cliques of a graph, lists all the sets of vertices of size at most $n/3$ and then, by applying a polynomial time procedure for each set, generates all the potential maximal cliques of the input graph.

Lemma 11. *Let Ω be a potential maximal clique of G, S be a minimal separator contained in Ω and C be the component of $G \setminus S$ intersecting Ω. Then one of the following holds:*

1. *$\Omega = N(C \setminus \Omega)$;*
2. *there is $a \in \Omega \setminus S$ such that $\Omega = N[a]$;*
3. *there is $a \in S$ such that $\Omega = S \cup (N(a) \cap C)$.*

Proof. Since C is a component of $G \setminus S$ and S is contained in Ω, we have that $N(C \setminus \Omega) \subseteq \Omega$. If every vertex of Ω is adjacent to a vertex of $C \setminus \Omega$, then $\Omega = N(C \setminus \Omega)$.

Suppose that there is a vertex $a \in \Omega$ having no neighbor in $C \setminus \Omega$. We consider first the case $a \in \Omega \setminus S$. We claim that in this case $\Omega = N[a]$. Because $a \in \Omega \setminus S \subseteq C$ we conclude that $N[a] \subseteq \Omega$. Thus to prove the claim we need to show that $\Omega \subseteq N[a]$. For sake of contradiction, suppose that there is $b \in \Omega$ which is not adjacent to a. By Theorem 4, every two non adjacent vertices of a potential maximal clique are contained in some minimal separator $S_i(\Omega)$. Thus both a and b should have neighbors in a component $C_i(\Omega)$ of $G \setminus \Omega$. Since $a \in \Omega \setminus S \subseteq C$, we have that $C_i(\Omega) \subseteq C \setminus \Omega$. But this contradicts the assumption that a has no neighbors in $C \setminus \Omega$.

The case $a \in S$ is similar. Suppose that $\Omega \setminus S \neq N(a) \cap C$, i.e. there is a vertex $b \in \Omega \setminus S$ non adjacent to a. Then again, a and b are contained in some minimal separator and thus should have neighbors in a component $C_i(\Omega) \subseteq C$ of $G \setminus \Omega$ which is a contradiction. $\square$

Definition 12. Let Ω be a potential maximal clique of G. The triple (S, a, b) is called a *separator representation* of Ω if S is a minimal separator of G, $a \in S$, $b \in V \setminus S$ and $\Omega = S \cup (N(a) \cap C_b(S))$, where $C_b(S)$ is the component of $G \setminus S$ containing b.

The number of all possible separator representations of a graph is at most $n^2|\Delta_G|$. Unfortunately, not every potential maximal clique has a separator representation. In the next subsection we introduce two different types of representations, the *partial representation* and the *indirect representation*, that allows us to show that all the potential maximal cliques can be represented by small sets of vertices.

5.1 Upper bounding the number of nice potential maximal cliques

Definition 13. Let Ω be a potential maximal clique of a graph G and let $S \subset \Omega$ be a minimal separator of G. We say that S is an *active separator for* Ω if Ω is not a clique in the graph $G_{\mathcal{S}(\Omega)\setminus\{S\}}$, obtained from G by completing all the minimal separators contained in Ω, except S. If S is active, a pair of vertices $x, y \in S$ non adjacent in $G_{\mathcal{S}(\Omega)\setminus\{S\}}$ is called an *active pair*. Otherwise, S is called *inactive for* Ω.

Theorem 14 ([10]). *Let Ω be a potential maximal clique of G and $S \subset \Omega$ a minimal separator, active for Ω. Let (S, C) be the block associated to S containing Ω and let $x, y \in \Omega$ be an active pair. Then $\Omega \setminus S$ is a minimal x, y-separator in $G[C \cup \{x, y\}]$.*

Definition 15. We say that a potential maximal clique Ω is *nice* if at least one of the minimal separators contained in Ω is active for Ω.

We shall prove first that a graph with n vertices has $\mathcal{O}^*(\binom{n}{n/3})$ nice potential maximal cliques.

Lemma 16. *Let Ω be a nice potential maximal clique, S be a minimal separator active for Ω, $x, y \in S$ be an active pair, and C be the component of $G\setminus S$ containing $\Omega \setminus S$. There is a partition (D_x, D_y, D_r) of $C \setminus \Omega$ such that $N(D_x \cup \{x\}) \cap C = N(D_y \cup \{y\}) \cap C = \Omega \setminus S$.*

Proof. By Theorem 14, $\Omega \setminus S$ is a minimal x, y-separator in $G[C \cup \{x, y\}]$. Let C_x be the full component associated to $\Omega \setminus S$ in $G[C \cup \{x, y\}]$ containing x, $D_x = C_x \setminus \{x\}$, and let C_y be the full component associated to $\Omega \setminus S$ in $G[C \cup \{x, y\}]$ containing y, $D_y = C_y \setminus \{y\}$, and $D_r = C \setminus (\Omega \cup D_x \cup D_y)$. Since $D_x \cup \{x\}$ and $D_y \cup \{y\}$ are full components of $\Omega \setminus S$, we have that $N(D_x \cup \{x\}) \cap C = N(D_y \cup \{y\}) \cap C = \Omega \setminus S$. □

Definition 17. For a potential maximal clique Ω of G, we say that a pair (X, c), where $X \subset V$ and $c \in X$ is a *partial representation* of Ω if $\Omega = N(C_c) \cup (X \setminus C_c)$, where C_c is the connected component of $G[X]$ containing c.

Definition 18. For a potential maximal clique Ω of G, we say that a triple (X, x, c), where $X \subset V$ and $x, c \notin X$ is an *indirect representation* of Ω if $\Omega = N(C_c \cup D_x \cup \{x\}) \cup \{x\}$, where

- C_c is the connected component of $G \setminus N[X]$ containing c;
- D_x is the vertex set of the union of all connected components C' of $G[X]$ such that $x \in N(C')$.

Let us note that for a given vertex set X and two vertices x, c one can check in polynomial time whether the pair (X, c) is a partial representation or if the triple (X, x, c) is a separator representation or indirect representation of a (unique) potential maximal clique Ω.

We state now the main tool for upper bounding the number of nice potential maximal cliques.

Lemma 19. *Let Ω be a nice potential maximal clique of G. Then one of the following holds:*

1. *There is a vertex a such that $\Omega = N[a]$;*
2. *Ω has a separator representation;*
3. *Ω has a partial representation (X, c) such that $|X| \leq n/3$;*
4. *Ω has a indirect representation (X, x, c) such that $|X| \leq n/3$.*

Proof. Let S be a minimal separator active for Ω, $x, y \in S$ be an active pair, and C be the component of $G \setminus S$ containing $\Omega \setminus S$. By Lemma 16, there is a partition (D_x, D_y, D_r) of $C \setminus \Omega$ such that $N(D_x \cup \{x\}) \cap C = N(D_y \cup \{y\}) \cap C = \Omega \setminus S$. If one of the sets D_x, D_y, say D_x, is equal the emptyset, then $N(D_x \cup \{x\}) \cap C = N(x) \cap C = \Omega \setminus S$, and thus the triple (S, x, z) is a separator representation of Ω.

Suppose that none of the first two conditions of the lemma holds. Then D_x and D_y are nonempty. In order to argue that Ω has a partial representation (X, c) or a indirect representation (X, x, c) such that $|X| \leq n/3$, we partition the graph further. Let $R = \Omega \setminus S$ and let D_S be the union of all full components associated to S in $G \setminus \Omega$. The vertex set D_x is the union of vertex sets of all connected components C' of $G \setminus (\Omega \cup D_S)$ such that x is contained in the neighborhood of C'. Thus a connected component C' of $G \setminus (\Omega \cup D_S)$ is contained in D_x if and only if $x \in N(C')$. Similarly, a connected component C' of $G \setminus (\Omega \cup D_S)$ is contained in D_y if and only if $y \in N(C')$. We also define $D_r = V \setminus (\Omega \cup D_S \cup D_x \cup D_y)$, which is the set of vertices of the components of $G \setminus (\Omega \cup D_S)$ which are not in D_x and D_y.

We partition S in the following sets

- $S_{\overline{x}} = (S \setminus N(D_x)) \cap N(D_y)$;
- $S_{\overline{y}} = (S \setminus N(D_y)) \cap N(D_x)$;
- $S_{\overline{xy}} = S \setminus (N(D_y) \cup N(D_x))$;
- $S_{xy} = S \cap N(D_y) \cap N(D_x)$.

Thus $S_{\overline{x}}$ is the set of vertices in S with no neighbor in D_x and with at least one neighbor in D_y, $S_{\overline{y}}$ is the set of vertices in S with no neighbor in D_y and with at least one neighbor in D_x, $S_{\overline{xy}}$ is the set of vertices in S with neighbors neither in D_x or D_y, and finally S_{xy} is the set of vertices in S with neighbors both in D_x and D_y. Notice that the vertex sets D_S, D_x, D_y, D_r, R,$S_{\overline{x}}$, $S_{\overline{y}}$, $S_{\overline{xy}}$, and S_{xy} are pairwise disjoint. The set S_{xy} is only mentioned to complete the partition of S, and will not be used in the rest of the proof.

Both for size requirements and because of the definition of indirect representation we can not use the sets $S_{\overline{x}}, S_{\overline{y}}$, and $S_{\overline{xy}}$ directly, they have to be represented by the sets $Z_{\overline{x}}, Z_{\overline{y}}$, and $Z_{\overline{r}}$, which are subsets of the vertex sets D_y, D_x, and D_r. By the definition of $S_{\overline{x}}$ and $S_{\overline{y}}$ it follows that there exists two vertex sets $Z_{\overline{x}} \subseteq D_y$ and $Z_{\overline{y}} \subseteq D_x$ such that $S_{\overline{x}} \subseteq N(Z_{\overline{x}})$ and $S_{\overline{y}} \subseteq N(Z_{\overline{y}})$, let $Z_{\overline{x}}$ and $Z_{\overline{y}}$ be the smallest such sets. By Lemma 11, $\Omega = N(D_x \cup D_y \cup D_r)$, thus it follows that there exists a vertex set $Z_{\overline{r}} \subseteq D_r$ such that $S_{\overline{xy}} \subseteq N(Z_{\overline{r}})$, let $Z_{\overline{r}}$ be the smallest such set. A sketch of how these vertex sets relates to each other is given in Figure 1.

Let C^* be a connected component of $G[D_S]$, remember that $N(C^*) = S$. We define the following sets

- $X_1 = C^* \cup R$;
- $X_2 = D_x \cup Z_{\overline{x}} \cup Z_{\overline{r}}$;
- $X_3 = D_y \cup Z_{\overline{y}} \cup Z_{\overline{r}}$.

First we claim that

- the pair (X_1, c), where $c \in C^*$, is a partial representation of Ω;

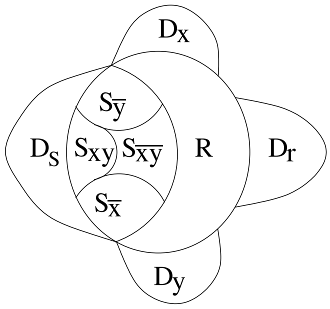

Figure 1: The figure shows a sketch of how the vertex sets $D_S, D_x, D_y, D_r, R, S_{\overline{x}}, S_{\overline{y}}, S_{\overline{xy}}$, and S_{xy} partition the graph G, and how the sets $Z_{\overline{x}}$, $Z_{\overline{y}}$, and $Z_{\overline{r}}$ relates to this partition.

- the triple (X_2, x, c), where $c \in C^*$ is an indirect representation of Ω;
- the triple (X_3, x, c), where $c \in C^*$ is an indirect representation of Ω.

In fact, the pair $(X_1, c) = (C^* \cup R, c)$ is a partial representation of Ω because $N(C^*) \cap R = \emptyset$, C^* induces a connected graph, and $\Omega = N(C^*) \cup R$. Thus (X_1, c) is a partial representation of Ω.

To prove that $(X_2, x, c) = (D_x \cup Z_{\overline{x}} \cup Z_{\overline{r}}, x, c)$ is an indirect representation of Ω, we have to show that $\Omega = N(C_c \cup D'_x \cup \{x\}) \cup \{x\}$ where C_c is the connected component of $G \setminus N[X_2]$ containing c, and D'_x is the vertex set of the union of all connected components C' of $G[X_2]$ such that $x \in N(C')$. Notice that $(S \cup C^*) \cap X_2 = \emptyset$ and that $S \subseteq N(X_2)$ since $S \subseteq N(D_x \cup Z_{\overline{x}} \cup Z_{\overline{r}})$ and $X_2 = D_x \cup Z_{\overline{x}} \cup Z_{\overline{r}}$. Hence the connected component C_c of $G \setminus N[X_2]$ containing c is C^*.

Every connected component C' of $G[X_2]$ is contained in $D_x, Z_{\overline{x}}$, or $Z_{\overline{r}}$ since $\Omega \cap (D_x \cup Z_{\overline{x}} \cup Z_{\overline{r}}) = \emptyset$ and Ω separates $D_x, Z_{\overline{x}}$, and $Z_{\overline{r}}$. From the definition of D_x it follows that $x \in N(C')$ for every component C' of $G[D_x]$, and from the definition of D_y and D_r follows that $x \notin N(C')$ for every component C' of $G[Z_{\overline{x}} \cup Z_{\overline{r}}]$. We can now conclude that D_x is the vertex set of the union of all connected components C' of $G[X_2]$ such that $x \in N(C')$. It remains to prove that $\Omega = N(C^* \cup D_x \cup \{x\}) \cup \{x\}$. By Lemma 16, we have that $\Omega \setminus S = R$ is subset of $N(D_x \cup \{x\})$ and $N(D_y \cup \{y\})$, and remember that $N(C^*) = S$. From this observations it follows that $\Omega = N(C^* \cup D_x \cup \{x\}) \cup \{x\}$ since $N(C^* \cup D_x \cup \{x\}) = (S \cup R) \setminus \{x\}$.

By similar arguments, (X_3, x, c) is an indirect representation of Ω.

To conclude the proof of Lemma, we argue that at least one of the vertex sets X_1, X_2, or X_3 used to represent Ω, contains at most $n/3$ vertices.

We partition the graph in the following three sets:

- $V_1 = D_S \cup R$;
- $V_2 = D_x \cup S_{\overline{x}} \cup S_{\overline{xy}}$;
- $V_3 = D_y \cup S_{\overline{y}} \cup D_r$.

These sets are pairwise disjoint and at least one of them is of size at most $n/3$ and to prove the Lemma we show that $|X_1| \leq |V_1|$, $|X_2| \leq |V_2|$, and $|X_3| \leq |V_3|$.

$\underline{|X_1| \leq |V_1|}$. Since $C^* \subseteq D_S$, we have that $X_1 = C^* \cup R \subseteq V_1 = D_S \cup R$.

$\underline{|X_2| \leq |V_2|}$. To prove the inequality we need an additional result

$$|Z_{\overline{x}}| \leq |S_{\overline{x}}|, |Z_{\overline{y}}| \leq |S_{\overline{y}}|, \text{ and } |Z_{\overline{r}}| \leq |S_{\overline{xy}}|. \tag{1}$$

In fact, since $Z_{\overline{x}}$ is the smallest subset of D_y such that $S_{\overline{x}} \subseteq N(Z_{\overline{x}})$, we have that for any vertex $u \in Z_{\overline{x}}$, $S_{\overline{x}} \not\subseteq N(Z_{\overline{x}} \setminus \{u\})$. Thus u has a private neighbor in $S_{\overline{x}}$, or in other words there exists $v \in S_{\overline{x}}$ such that $\{u\} = N(v) \cap Z_{\overline{x}}$. Therefore $S_{\overline{x}}$ contains at least one vertex for every vertex in $Z_{\overline{x}}$, which yields $|Z_{\overline{x}}| \leq |S_{\overline{x}}|$. The proof of inequalities $|Z_{\overline{y}}| \leq |S_{\overline{y}}|$, and $|Z_{\overline{r}}| \leq |S_{\overline{xy}}|$ is similar.

Now the proof of $|X_2| \leq |V_2|$, which is equivalent to $|D_x \cup Z_{\overline{x}} \cup Z_{\overline{r}}| \leq |D_x \cup S_{\overline{x}} \cup S_{\overline{xy}}|$, follows from (1) and the fact that all subsets on each side of inequality are pairwise disjoint.

$\underline{|X_3| \leq |V_3|}$. This inequality is equivalent to $|D_y \cup Z_{\overline{y}} \cup Z_{\overline{r}}| \leq |D_y \cup S_{\overline{y}} \cup D_r|$. Again, the sets on each side of inequality are pairwise disjoint. $|Z_{\overline{r}}| \leq |D_r|$ because $Z_{\overline{r}} \subseteq D_r$, and $|Z_{\overline{y}}| \leq |S_{\overline{y}}|$ by (1).

Thus $\min\{|X_1|, |X_2|, |X_3|\} \leq n/3$ which concludes the proof of the lemma. □

Lemma 20. *Every graph on n vertices has at most $2n^2 \sum_{i=1}^{n/3} \binom{n}{i}$ nice potential maximal cliques which can be listed in $\mathcal{O}^*(\binom{n}{n/3})$ time.*

Proof. By Lemma 19, the number of the number of possible partial representations (X, c) and indirect representations (X, x, c) with $|X| \leq n/3$ is at most $2n^2 \sum_{i=1}^{n/3} \binom{n}{i}$. By Theorem 10, the number of all possible separator representations is at most $n^2|\Delta_G| \leq n^2\binom{n}{n/3}$ and we deduce that the number of nice potential maximal cliques is at most $2n^2 \sum_{i=1}^{n/3} \binom{n}{i}$. Moreover, these potential maximal cliques can be computed in $\mathcal{O}^*(\binom{n}{n/3})$ time as follows. We enumerate all the triples (S, a, b) where S is a minimal separator and a, b are vertices, and check if the triple is the separator representation of a potential maximal clique Ω; if so, we store this potential maximal clique. We also enumerate all the potential maximal cliques of type $N[a], a \in V(G)$ in polynomial time. Finally, by listing all the sets X of at most $n/3$ vertices and all the couples of vertices (x, c), we compute all the nice potential maximal cliques with a partial representation (X, c) or a indirect representation (X, x, c). □

5.2 Upper bounding the number of potential maximal cliques

Not all potential maximal cliques of a graph are necessarily nice (see [10] for an example). For upper bounding and listing all potential maximal cliques of a graph, we need the following theorem, used in [10] for showing that the number of potential maximal cliques of G is $\mathcal{O}^*(|\Delta_G|^2)$.

Theorem 21 ([10]). *Let Ω be a potential maximal clique of G, let a be a vertex of G and $G' = G \setminus \{a\}$. Then one of the following cases holds:*

1. *either Ω or $\Omega \setminus \{a\}$ is a potential maximal clique of G'.*
2. *$\Omega = S \cup \{a\}$, where S is a minimal separator of G.*
3. *Ω is nice.*

Theorem 22. *A graph G on n vertices has at most $2n^3 \sum_{i=1}^{n/3} \binom{n}{i} = \mathcal{O}(n^4 \cdot 1.8899^n)$ potential maximal cliques. There is an algorithm to list all potential maximal cliques of a graph in time $\mathcal{O}^*(1.8899^n)$.*

Proof. Let $x_1, x_2, \ldots, x_n$ be the vertices of G and $G_i = G[\{x_1, \ldots, x_i\}]$, for all $i \in \{1, 2, \ldots, n\}$. Theorem 21 and Lemma 20 imply that $|\Pi_{G_i}| \leq |\Pi_{G_{i-1}}| + n|\Delta_{G_i}| + 2n^2 \sum_{i=1}^{n/3} \binom{n}{i}$, for all $i \in \{2, 3, \ldots, n\}$. By Theorem 10, $|\Pi_G| \leq 2n^3 \sum_{i=1}^{n/3} \binom{n}{i}$.

Clearly, if we have the potential maximal cliques of G_{i-1}, the potential maximal cliques of G_i can be computed in $\mathcal{O}^*(|\Pi_{G_{i-1}}| + \binom{n}{n/3})$ time by making use of Theorems 21, 10, and Lemma 20. The graph G_1 has a unique potential maximal clique, namely $\{x_1\}$. Therefore Π_G can be listed in time $\mathcal{O}^*(\binom{n}{n/3})$ time which is approximately $\mathcal{O}^*(1.8899^n)$. $\square$

Theorems 8 and 22 imply the main result of this paper.

Theorem 23. *For a graph G on n vertices, the treewidth and the minimum fill-in of G can be computed in $\mathcal{O}^*(1.8899^n)$ time.*

6 AT-free graphs

In this section we establish exact algorithms to compute the treewidth and the minimum fill-in of asteroidal-triple free graphs which are faster than the ones obtained for general graphs in the previous section. Both algorithms are based on new upper bounds on the number of minimal separators and the number of potential maximal cliques in AT-free graphs.

Three pairwise non-adjacent vertices of a graph G form an *asteroidal triple* (AT for short) if any two of them are connected by a path avoiding the neighborhood of the third vertex. Graphs without asteroidal triples are called *AT-free.*

Corneil, Olariu & Stewart studied structural properties of AT-free graphs in their fundamental paper [18]. Among others they showed that every connected AT-free graph has a dominating pair, where two vertices x and y of G form a *dominating pair* (DP for short) if the vertex set of each x, y-path is a dominating set of G.

AT-free graphs contain cocomparability graphs, permutation graphs, interval graphs and cobipartite graphs. Thus the treewidth problem and the minimum fill-in problem remain NP-hard when restricted to AT-free graphs [2, 50].

Remark. There is a well-known cobipartite (and thus AT-free) graph consisting of two cliques of size $n/2$ and a perfect matching between them which has precisely $2^{n/2} - 2$ minimal separators. It is not hard to show that this is indeed the largest number of minimal separators of a cobipartite graph on n vertices.

In a first part we show that $|\Pi_G| = \mathcal{O}^*(|\Delta_G|)$ for AT-free graphs, improving a result in [9] (Corollary 5.2). This also establishes an algorithm to list the potential maximal cliques of an AT-free graph in $\mathcal{O}^*(|\Delta_G|)$ time. In a second part we prove that an AT-free graph on n vertices has at most $2^{n/2+3}$ minimal separators.

First let us summarize some structural properties of potential maximal cliques in AT-free graphs.

Lemma 24 (Proposition 5.1 [9]). *Let Ω be a potential maximal clique of an AT-free graph G. Then the set $\mathcal{S}(\Omega)$ of minimal separators contained in Ω has at most two inclusion-maximal elements.*

Lemma 25 (Theorem 3.10 [9]). *Let G be a graph and Ω be a potential maximal clique of G such that $\mathcal{S}(\Omega)$ has a unique inclusion-maximal element S. Then $\Omega \setminus S$ is a connected component of $G \setminus S$.*

Let S and T be two non-crossing minimal separators of G, incomparable with respect to inclusion. Thus S meets a unique component of $G \setminus T$ say $C_S(T)$, and T meets a unique component of $G \setminus S$ say $C_T(S)$. We define the *piece between S and T* as $P(S,T) = S \cup T \cup (C_T(S) \cap C_S(T))$.

Lemma 26 (Theorem 3.11 [9]). *Let G be a graph and Ω be a potential maximal clique of G such that $\mathcal{S}(\Omega)$ has exactly two inclusion-maximal elements S and T. Then $\Omega = P(S,T)$.*

Lemma 27. *Let G be an AT-free graph and Ω be a potential maximal clique of G such that $\mathcal{S}(\Omega)$ has two inclusion-maximal elements S and T. Choose $s \in S \setminus T$. Then $\Omega = S \cup (N(s) \cap C_T(S))$.*

Proof. By Lemma 26, $\Omega = P(S,T)$. Clearly s is in the unique component $C_S(T)$ of $G \setminus T$ meeting S, so $N(s) \cap C_T(S) \subseteq P(S,T)$. Consequently $S \cup (N(s) \cap C_T(S)) \subseteq \Omega$.

Conversely, suppose there is a vertex $t \in \Omega$, not contained in $S \cup (N(s) \cap C_T(S))$. Let $S' = (S \setminus \{s\}) \cup (N(s) \cap C_T(S))$. Clearly S' separates s and any vertex of $C_T(S) \setminus S'$ in G, in particular S' separates s and t. It follows that there is a minimal separator $S'' \subseteq S'$ of G, contained in Ω and separating two vertices of Ω. According to Theorem 4, for each minimal separator U contained in Ω, Ω intersects exactly one component of $G \setminus U$, which is a contradiction. □

Theorem 28. *An AT-free graph G has at most $n^2|\Delta_G| + n|\Delta_G| + 1$ potential maximal cliques. Furthermore, there is an algorithm to list the potential maximal cliques of an AT-free graph in $\mathcal{O}^*(|\Delta_G|)$ time.*

Proof. If G has no minimal separator, then G is a complete graph and its vertex set is the unique potential maximal clique of G.

Suppose now that G is not complete. Fix a minimal separator S of G. By Lemma 25, the number of potential maximal cliques Ω such that S is the unique inclusion-maximal element of $\mathcal{S}(\Omega)$ is bounded by the number of connected components of $G \setminus S$. Hence, there are at most n such potential maximal cliques.

Now let us consider the potential maximal cliques Ω for which S is one of the two inclusion-maximal separators contained in $\mathcal{S}(\Omega)$. For any component C of $G \setminus S$, there are, by Lemma 27, at most $|S|$ such potential maximal cliques contained in $S \cup C$. It follows that there are at most n^2 potential maximal cliques of this type.

Therefore, G contains at most $(n^2 + n)|\Delta_G| + 1$ potential maximal cliques. These combinatorial arguments can easily be transformed into an algorithm listing the potential maximal cliques of an AT-free graph in time $O^*(|\Delta_G|)$. □

Hence Theorem 8 implies that to construct an $O^*(1.4142^n)$ algorithm computing the tree-width and the minimum fill-in of an AT-free graph it is enough to prove that the number of minimal separators in an AT-free graph is $O^*(1.4142^n)$.

Our proof that the number of minimal separators in an AT-free graph is at most $2^{n/2+3}$ relies on properties of 2LexBFS, i.e. a combination of two runs of lexicographic breadth-first-search (also called 2-sweep LexBFS), on AT-free graphs established by Corneil, Olariu & Stewart in [19].

Definition 29. A vertex ordering $x_n, x_{n-1}, \dots, x_1$ is said to be a *2LexBFS ordering* of G if some 2LexBFS(G) returns the vertices in this order (starting with x_n) during the second sweep of LexBFS on G where x_n is supposed to be the last vertex of the first sweep of LexBFS on G.

We shall write $u \prec v$ if $u = x_i$, $v = x_j$ and $i < j$. A 2LexBFS ordering and the levels $L_0 = \{x_n\}, L_1 = N(x_n), \dots, L_i = \{x_j : d(x_j, x_n) = i\}, \dots, L_r$ are called a *2LexBFS scheme* of G. Consider any 2LexBFS scheme. Clearly all neighbors of a vertex $v \in L_i$ are contained in $L_{i-1} \cup L_i \cup L_{i+1}$. For a vertex $v \in L_i$ we denote $N(v) \cap L_{i-1}$ by $N^{\uparrow}(v)$, and we denote $N(v) \cap L_{i+1}$ by $N_{\downarrow}(v)$.

Theorem 30 ([19]). *Every 2LexBFS ordering $x_n, x_{n-1}, \dots, x_1$ of a connected AT-free graph has the* dominating pair-property *(DP-property for short), i.e., for all $i \in \{1, 2, \dots, n\}$, (x_n, x_i) is a dominating pair of the graph $G[\{x_i, x_{i+1}, \dots, x_n\}]$.*

The following easy consequence of Theorem 30 is useful.

Lemma 31. *Let $x_n, x_{n-1}, \dots, x_1$ be a 2LexBFS ordering of an AT-free graph G and let $L_0, L_1, \dots, L_r$ be the corresponding 2LexBFS scheme. Let $s > r$, $x_s, x_r \in L_i$ and $\{x_r, x_s\} \notin E$. Then $N^{\uparrow}(x_r) \subseteq N^{\uparrow}(x_s)$.*

Proof. Let $w \in N^{\uparrow}(x_r) \setminus N^{\uparrow}(x_s)$. Then the path $x_r, w, u_{i-2}, \dots, u_1, x_n$ with $u_j \in L_j$ and $u_{j-1} \in N^{\uparrow}(u_j)$ for all $j = i-2, \dots, 1$ contains no neighbor of x_s contradicting the DP-property of a 2LexBFS scheme of an AT-free graph. □

Theorem 32. *An AT-free graph on n vertices has at most $2^{n/2+3}$ minimal separators.*

Proof. Let G be an AT-free graph. Let $x_n, x_{n-1}, \dots, x_1$ be a 2LexBFS ordering of G and let $L_0, L_1, \dots, L_r$ be the levels of the corresponding 2LexBFS scheme.

Let S be any minimal separator of G. Let C and C' be two (not necessarily full) components of $G \setminus S$. We claim that at most one level of the 2LexBFS scheme may contain vertices of C and C'. Suppose not. Let L_i and L_{i+1} be levels containing vertices of C and C'. Then there are edges $\{u, v\}$ in C and $\{w, x\}$ in C' such that $u, w \in L_i$ and $v, x \in L_{i+1}$. W.l.o.g. assume $u \prec w$. Then Lemma 31 implies that w and v are adjacent, a contradiction.

Let C and C' be two (not necessarily full) components of $G \setminus S$ such that both contain vertices of some level of the 2LexBFS scheme, say L_i. Furthermore assume $C \cap L_{i-1} \neq \emptyset$ and $C' \cap L_{i-1} = \emptyset$. Hence there is an edge $\{u, v\}$ in C such that $u \in L_i$ and $v \in L_{i-1}$. Then for each $w \in C'$ holds $w \prec u$. Otherwise $u \prec w$, $w \in L_i$ and Lemma 31 would imply that w and v are adjacent, a contradiction.

Finally we claim that in this case $c' \prec c$ for each vertex $c \in C$ and each vertex $c' \in C'$. This is obviously true if one of c and c' is not in L_i. It remains to consider the case $c \in L_i$, $c' \in L_i$. To the contrary assume $c \prec c'$. Since C contains vertices of L_i and L_{i-1} there is a path in C starting in c passing through vertices of $C \cap L_i$ only until it passes through an edge $\{u, v\}$ in C with $u \in L_i$ and $v \in L_{i-1}$. This path can be extended to a path from c to x_n that does not contain a neighbor of c' although $c \prec c'$, a contradiction to the DP-property.

Now we are able to upper bound the number of those minimal separators in an AT-free graph in which no full component contains only vertices of one level. Simply divide the vertex set into two halves: $A = \{x_n, x_{n-1}, \dots, x_{\lceil n/2 \rceil + 1}\}$ and $B = \{x_{\lfloor n/2 \rfloor}, \dots, x_1\}$. Now consider two full components C and C' of a minimal separator S of G, i.e. $S = N(C) = N(C')$. Then either C or C' is a subset of either A or B, and surely each of them C and C' uniquely determines S. Hence we simply consider all subsets of A and all subsets of B as possible full components of a minimal separator of G. Consequently there are at most $2^{n/2+1}$ minimal separators of this type.

It remains to upper bound the number of all those minimal separators S of an AT-free graph G for which each full component is neither a subset of A nor a subset of B. Hence at least one full component of S contains only vertices from one level of the 2LexBFS scheme.

Let S be such a minimal separator of G. Let C and C' be two full components of $G \setminus S$. W.l.o.g. assume $C \subseteq L_i$. Hence $x_{\lfloor n/2 \rfloor} \in L_i$, and thus the level L_i is uniquely determined.

$C' \cap \bigcup_{j=0}^{i-1} L_j = \emptyset$ since otherwise $c \prec c'$ for all $c \in C$ and all $c' \in C'$, and either C or C' is a subset of A or B. Similarly C' must contain vertices of L_i. Consequently $C' \subseteq \bigcup_{j=i}^{r} L_j$. It is easy to see that $C \subseteq L_i$ and $S = N(C)$ imply $N(C') = S \subseteq \bigcup_{j=i-1}^{i+1} L_j$. Furthermore $N(C) = N(C') = S$ implies $S \cap L_{i-1} = N^{\uparrow}(C \cap L_i) = N^{\uparrow}(C' \cap L_i)$.

Now let us consider the graph $G' = G \setminus \bigcup_{j=0}^{i-1} L_j$. Then $S' = S \setminus \bigcup_{j=0}^{i-1} L_j$ is a separator of G'; C and C' are components of $G' \setminus S'$. Furthermore, every vertex of $S' \subseteq S$ has a neighbor in C and C', and thus S' is a minimal separator of G'. Consequently every minimal separator S of G for which no full component is a subset of A or B corresponds uniquely to a minimal separator of G'. Notice G' has at most $n-1$ vertices since we remove at least one vertex of L_{i-1} from G to obtain G'.

Let $f(n)$ be a function such that $f(n)$ is an upper bound for the number of minimal separators in an n-vertex AT-free graph. Then we established the recurrence $f(n) \geq 2^{n/2+1} + f(n-1)$ and conclude with $f(n) = 4 \cdot 2^{n/2+1} = 8 \cdot 2^{n/2}$. $\square$

Combining Theorems 8, 28, and 32 we obtain algorithms for AT-free graphs being faster than the corresponding ones for general graphs.

Theorem 33. *There are algorithms to compute the treewidth and the minimum fill-in of an AT-free graph in $O^*(1.4142^n)$ time.*

7 Open problems and final remarks

Planar graphs. The computational complexity of treewidth restricted to planar graphs is a long standing open problem in Graph Algorithms. The treewidth of planar graphs can be approximated within a constant factor of 1.5. More precisely, Seymour and Thomas [43] gave a polynomial algorithm for computing the *branchwidth* of planar graphs, and the latter parameter differs by at most a factor of 1.5 from the treewidth.

In the case of planar graphs with n vertices, the treewidth is at most $\mathcal{O}(\sqrt{n})$.

Theorem 34 ([27]). *For any planar graph G on n vertices,* $\mathrm{tw}(G) \leq 3.182\sqrt{n} + \mathcal{O}(1)$.

Also given a graph G and a number k, one can decide if $\mathrm{tw}(G) \leq k$ in $\mathcal{O}^*(n^k)$ time, either using the algorithm of Arnborg et al. [2] or using our technique, restricted to potential maximal cliques of size at most $k+1$.

Consequently the treewidth of planar graphs can be computed in time $\mathcal{O}^*(n^{3.182\sqrt{n}}) = 2^{\mathcal{O}(\sqrt{n}\log n)}$.

Unfortunately, although the structure of potential maximal cliques in planar graphs is very particular [12], our approach can not be used for obtaining algorithms of running time $2^{\mathcal{O}(\sqrt{n})}$ for planar treewidth. This is because the number of 'large' potential maximal cliques in planar graphs can be 'large'.

Claim 2. *For any integer N, there is a planar graph on $n > N$ vertices with at least $2^{0.49\sqrt{n}\log n}$ potential maximal cliques of size at least $2\sqrt{n}+2$.*

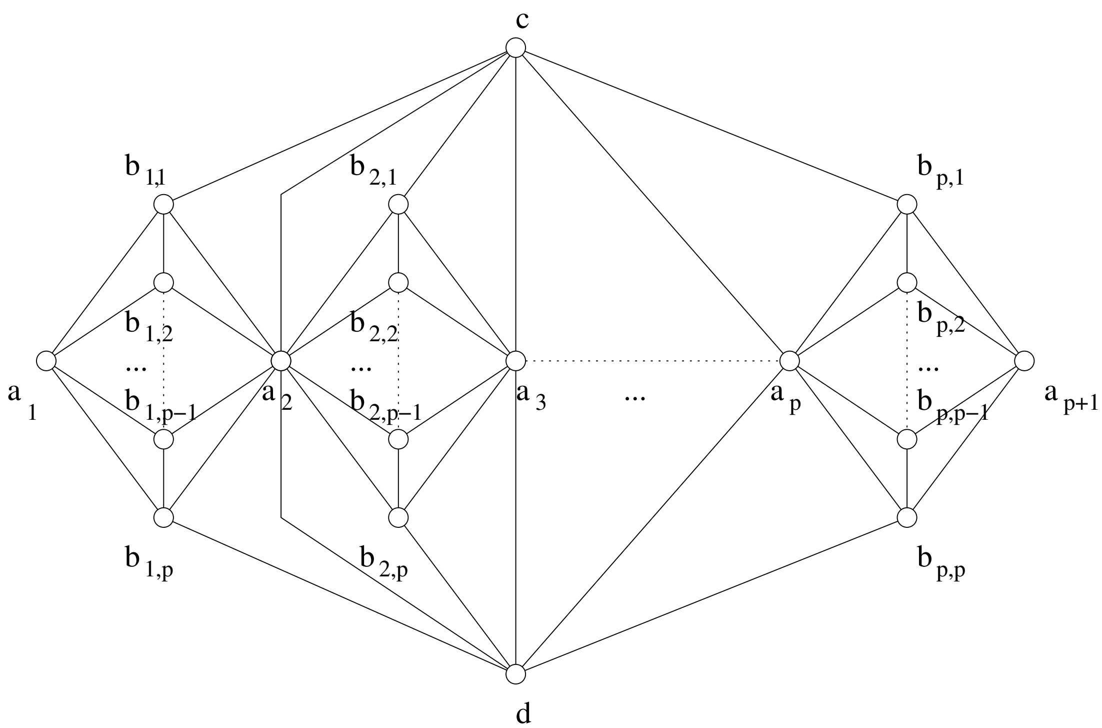

Figure 2: Planar graphs with many small potential maximal cliques

Proof. Consider the planar graph G_p depicted in Figure 2. It has $n = p^2 + p + 3$ vertices. The set of vertices $S = \{a_1, b_{1i_1}, a_2, b_{2i_2}, \ldots, a_p, b_{pi_p}, a_{p+1}\}$ forms a c,d-minimal separator for any values $i_1, i_2, \ldots, i_p$ between 1 and p. By making use of Theorem 4, it is not hard to see that $S \cup \{c\}$ is a potential maximal clique of size $p+1$ in G_p. Consequently, G has at least p^p potential cliques. If $p \geq 2$, we have $p > \sqrt{n} - 1$, thus the number of potential maximal cliques is at least $(\sqrt{n}-1)^{\sqrt{n}-1}$. $\square$

Since we do not know if the treewidth of a planar graph can be computed in polynomial time, an interesting task is to design an algorithm of running time $2^{\mathcal{O}(\sqrt{n})}$. As we mentioned, this will need new techniques.

Combinatorial bounds. The running time estimation of our algorithms is based on combinatorial upper bounds on minimal separators and a bound for listing all potential maximal cliques. To find better bounds on the number of minimal separators and potential maximal cliques are interesting combinatorial challenges.

How many potential maximal cliques can be in a graph? Recently Villanger [45] proved that the number of potential maximal cliques in a graph on n vertices is at most $n^3 \cdot 1.8135^n$. Unfortunately, it is not clear if the proof of Villanger can be turned into algorithm listing all potential maximal cliques in time $\mathcal{O}^*(1.8135^n)$. Of course, such an algorithm can be used to speed up our algorithm for treewidth and fill-in. A related interesting question is if it is possible to list potential maximal cliques with polynomial delay.

How many minimal separators can be in a graph? We are aware of the following construction providing the lower bound $3^{n/3} \approx 1.4422^n$ on the number of minimal separators: Let G be a graph on $n = 3k+2$ vertices. G has two vertices a, b that are connected by k vertex disjoint paths of length 4. Every minimal a,b-separator in G contains exactly one inner vertex of each a,b-path. So the number of minimal separators in G is at least $3^{n/3} \approx 1.4422^n$. However the gap between the lower bound and the upper bound $\mathcal{O}(n \cdot 1.7087^n)$ from Theorems 10 is still big.

For some special graph classes, the use of minimal separators can imply faster algorithms for triangulation problems. For example, we have shown that every AT-free graph on n vertices has at most $2^{n/2+3}$ minimal separators and that this upper bound is tight up to a multiplicative constant factor. The interesting question here is whether similar techniques can be used for other graph classes, like bipartite graphs and graphs of small degree.

More generally, it is a natural and challenging question to ask how many subgraphs satisfying a given property can be in a given graph. Surprisingly, despite the question is so natural, there are not so many known results of this type. For example, the number of perfect matchings in a simple k-regular bipartite graph on $2n$ vertices is always between $n!(k/n)^n$ and $(k!)^{n/k}$. (The first inequality was known as van der Waerden Conjecture [46] and was proved in 1980 by Egorychev [20] and the second is due to Bregman [13].) Another example is the famous Moon and Moser [36] theorem stating that every graph on n vertices has at most $3^{n/3}$

maximal cliques (independent sets). Byskov and Eppstein [16] obtain a 1.7724^n upper bound on the number of maximal bipartite subgraphs in a graph. Such combinatorial bounds are of interests not only on their own but also because often they are used for algorithm design as well.

Related problems. Our algorithms for treewidth and minimum fill-in can also be used for solving other problems that can be expressed in terms of minimal triangulations like finding a tree decomposition of minimum cost [8] or computing treewidth of weighted graphs. However, there are two 'width' parameters related to treewidth, namely bandwidth and pathwidth and one parameter called profile, related to minimum fill-in, that do not fit into this framework. Bandwidth can be computed in time $\mathcal{O}^*(10^n)$ [22] and reducing Feige's bounds is a challenging problem. Pathwidth (and profile) can be expressed as vertex ordering problems and thus solved in $\mathcal{O}^*(2^n)$ time by applying a dynamic programming approach similar to Held and Karp's approach [29] for TSP. Let us note that reaching time complexity $\mathcal{O}^*(c^n)$, for any constant $c < 2$ even for the Hamiltonian cycle problem is a long standing problem. So it is unlikely that some modification of Held & Karp's approach provide us with a better exact algorithm for pathwidth or profile. It is tempting to ask if one can reach time complexity $\mathcal{O}^*(c^n)$, for any constant $c < 2$ for these problems.

References

[1] E. Amir, *Efficient approximation for triangulation of minimum treewidth*, in Uncertainty in Artificial Intelligence: Proceedings of the Seventeenth Conference (UAI-2001), San Francisco, CA, 2001, Morgan Kaufmann Publishers, pp. 7–15.

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

[3] R. Beigel and D. Eppstein, *3-coloring in time $O(1.3289^n)$*, Journal of Algorithms, 54 (2005), pp. 168–204.

[4] A. Berry, J.P. Bordat, and O. Cogis, *Generating all the minimal separators of a graph*, Proceedings of the 25th Workshop on Graph-theoretic Concepts in Computer Science (WG'99), LNCS vol. 1665, Springer, Berlin, 1999, pp. 167–172.

[5] H. L. Bodlaender, *A linear-time algorithm for finding tree-decompositions of small treewidth*, SIAM J. Computing, 25 (1996), pp. 1305–1317.

[6] H. L. Bodlaender, *A partial k-arboretum of graphs with bounded treewidth*, Theoret. Comput. Sci., 209 (1998) , pp. 1–45.

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

[8] H. L. Bodlaender and F. V. Fomin, *Tree decompositions with small cost,* Proceedings of the 8th Scandinavian Workshop on Algorithm Theory (SWAT 2002), LNCS vol. 2368, Springer, Berlin, 2002, pp. 378–387.

[9] V. Bouchitté and I. Todinca, *Treewidth and minimum fill-in: grouping the minimal separators,* SIAM J. Computing, 31 (2001), pp. 212–232.

[10] V. Bouchitté and I. Todinca, *Listing all potential maximal cliques of a graph,* Theoret. Comput. Sci., 276 (2002), pp. 17–32.

[11] V. Bouchitté, D. Kratsch, H. Müller, and I. Todinca, *On treewidth approximation,* Discr. Appl. Math., 136 (2004), pp. 183–196.

[12] V. Bouchitté, F. Mazoit, and I. Todinca, *Chordal embeddings of planar graphs*, Discr. Math., 273 (2003), pp. 85–102.

[13] L. M. Brègman, *Certain properties of nonnegative matrices and their permanents*, Doklady Akademii Nauk BSSR, 211 (1973), pp. 27–30.

[14] T. Brueggemann and W. Kern, *An improved determenistic local search algorithm for 3-SAT*, Theoret. Comput. Sci., 329 (2004), pp. 303-313.

[15] J.M. Byskov, *Enumerating maximal independent sets with applications to graph colouring*, Operations Research Letters, 32 (2004),pp. 547–556.

[16] M. Byskov and D. Eppstein, *An algorithm for enumerating maximal bipartite subgraphs*, manuscript, (2004).

[17] L. Cai, *Fixed-parameter tractability of graph modification problems for hereditary properties*, Information Processing Letters, 58 (1996), pp. 171–176.

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

[19] D.G. Corneil, S. Olariu and L. Stewart, *Linear time algorithms for dominating pairs in asteroidal triple-free graphs,* SIAM J. Computing, 28 (2000), pp. 1284–1297.

[20] G. P. Egorychev, *Proof of the van der Waerden conjecture for permanents*, Sibirsk. Mat. Zh., 22 (1981), pp. 65–71, 225.

[21] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, Springer-Verlag, New York, 1999.

[22] U. Feige, *Coping with the NP-hardness of the graph bandwidth problem*, Proceeding of the 7th Scandinavian Workshop on in Algorithm theory (SWAT 2000), LNCS vol. 1851, Springer, Berlin, 2000, pp. 10–19.

[23] U. Feige, M. Hajiaghayi, and J. R. Lee, *Improved approximation algorithms for minimum-weight vertex separators*, in Proceedings of the 37th annual ACM Symposium on Theory of computing (STOC 2005), New York, 2005, ACM Press, pp. 563–572.

[24] F. V. Fomin, P. Fraigniaud, and N. Nisse, *Nondeterministic Graph Searching: From Pathwidth to Treewidth*, Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science (MFCS 2005), Springer-Verlag Lecture Notes in Computer Science, to appear.

[25] F. V. Fomin, F. Grandoni, and D. Kratsch, *Measure and conquer: Domination – a case study*, in Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005), vol. 3580 of LNCS, Springer, Berlin, 2005, pp. 191–203.

[26] F. V. Fomin, D. Kratsch, and I. Todinca, *Exact (exponential) algorithms for treewidth and minimum fill-in*, Proceedings of the 31st International Colloquium on Automata, Languages and Programming, ICALP 2004, Turku, Finland, July 12-16, 2004, pp. 568–580.

[27] F. Fomin and D. Thilikos, *A simple and fast approach for solving problems on planar graphs*, Proceedings of the 21st Annual Symposium on Theoretical Aspects of Computer Science, LNCS vol. 2996, Springer, Berlin, 2004, pp. 56–67.

[28] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[29] M. Held and R. Karp, *A dynamic programming approach to sequencing problems*, J. Soc. Indust. Appl. Math., 10 (1962), pp. 196–210.

[30] E. Horowitz and S. Sahni, *Computing partitions with applications to the knapsack problem*, Journal of ACM, 21 (1974), pp. 277–292.

[31] K. Iwama, *Worst-case upper bounds for ksat*, Bulletin of the EATCS, 82 (2004), pp. 61–71.

[32] H. Kaplan, R. Shamir, and R. E. Tarjan, *Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs*, SIAM J. Computing, 28 (1999), pp. 1906–1922.

[33] T. Kloks, D. Kratsch, and J. Spinrad, *On treewidth and minimum fill-in of asteroidal triple-free graphs*, Theoret. Comput. Sci., 175 (1997), pp. 309–335.

[34] E. L. Lawler, *A note on the complexity of the chromatic number problem*, Information Processing Letters, 5 (1976), pp. 66–67.

[35] B. Monien and E. Speckenmeyer, *Solving satisfiability in less than 2^n steps* Discr. Appl. Math., 10 (1985), pp. 287–295.

[36] J. W. Moon and L. Moser, *On cliques in graphs*, Israel Journal of Mathematics, 3 (1965), pp. 23–28.

[37] A. Parra and P. Scheffler, *Characterizations and algorithmic applications of chordal graph embeddings,* Discr. Appl. Math., 79 (1997), pp. 171–188.

[38] N. Robertson and P. D. Seymour, *Graph minors. II. Algorithmic aspects of tree-width*, J. Algorithms, 7 (1986), pp. 309–322.

[39] N. Robertson and P. D. Seymour, *Graph minors. X. Obstructions to tree-decomposition*, J. Combin. Theory Ser. B, 52 (1991), pp. 153–190.

[40] J.M. Robson, *Algorithms for maximum independent sets*, J. Algorithms, 7 (1986), pp. 425–440.

[41] R. Schroeppel And A. Shamir, *A $T = O(2^{n/2})$, $S = O(2^{n/4})$ algorithm for certain NP-complete problems*, SIAM J. Computing, 10 (1981),pp. 456–464.

[42] Schöning, *Algorithmics in exponential time*, in Proceedings of the 22nd International Symposium on Theoretical Aspects of Computer Science (STACS 2005), vol. 3404 of LNCS, Springer, Berlin, 2005, pp. 36–43.

[43] P. Seymour and R. Thomas, *Call routing and the ratcatcher*, Combinatorica, 14 (1994), pp. 217–241.

[44] R.E. Tarjan and A.E. Trojanowski, *Finding a maximum independent set*, SIAM J. Computing, 6 (1977), pp. 537–546.

[45] Y. Villanger, *Improved exponential-time algorithms for treewidth and minimum fill-in*, In *LATIN*, Lecture Notes in Computer Science. Springer Verlag, 2006. To appear.

[46] B. van der Waerden, *Problem 45*, Jahresber. Deutsch. Math.-Verein., 35 (1926), p. 117.

[47] R. Williams, *A new algorithm for optimal constraint satisfaction and its implications*, Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP 2004), LNCS vol. 3142, Springer, Berlin, 2004, pp. 1227–1237.

[48] G. Woeginger, *Exact algorithms for NP-hard problems: A survey*, in Combinatorial Optimization - Eureka, you shrink!, vol. 2570 of LNCS, Springer-Verlag, Berlin, 2003, pp. 185–207.

[49] ——, *Space and time complexity of exact algorithms: Some open problems*, Proceeding of the 1st International Workshop on Parameterized and Exact Computation (IWPEC 2004), vol. 3162 of LNCS, Springer-Verlag, Berlin, 2004, pp. 281–290.

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