UNIVERSITE D'ORLEANS
Faculté des Sciences
LIFO

Laboratoire d'Informatique Fondamentale d'Orléans
4, rue Léonard de Vinci, BP 6759
F-45067 Orléans Cedex 2
FRANCE

Rapport de Recherche

www : http://www.univ-orleans.fr/SCIENCES/LIFO/

Exact (exponential) algorithms for treewidth and minimum fill-in

Dieter KRATSCH, LITA, Université de Metz, France
Fedor FOMIN, Department of Informatics, University of Bergen, Norway
Ioan TODINCA, LIFO, Université d'Orléans, France

Rapport N^o **2004-09**

Exact (exponential) algorithms for treewidth and minimum fill-in

Fedor V. Fomin[1], Dieter Kratsch[2], and Ioan Todinca[3]

[1] Department of Informatics, University of Bergen, N-5020 Bergen, Norway, fomin@ii.uib.no
[2] LITA, Université de Metz, 57045 Metz Cedex 01, France, kratsch@sciences.univ-metz.fr
[3] LIFO, Université d'Orléans, 45067 Orléans Cedex 2, France, Ioan.Todinca@lifo.univ-orleans.fr

Abstract. We show that for a graph G on n vertices its treewidth and minimum fill-in can be computed in $\mathcal{O}(\text{poly}(n) \cdot 1.9601^n)$ time. Our result is based on a combinatorial proof that the number of minimal separators in a graph is $\mathcal{O}(n \cdot 1.7087^n)$ and that the number of potential maximal cliques is $\mathcal{O}(n^4 \cdot 1.9601^n)$.

1 Introduction

The last few years have seen an emerging interest in fast exponential algorithms for NP-hard problems. There are several explanations to this interest. Today almost everyone does expect that there is no polynomial time algorithm solving an NP-hard problem. Trying different ways of avoiding intractability several approaches were proposed such as: approximation algorithms, randomized algorithms, heuristic methods etc. Each of these approaches has weak points like necessity of exact solutions, difficulty of approximation, limited power of the method itself and many others. All these obstacles encourage us to try a direct (and perhaps desperate) way of coping with NP-hardness: Design of exponential algorithms that are significantly faster than exhaustive search. 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. Nice examples of fast exponential algorithms are Eppstein's graph coloring algorithm with time complexity $\mathcal{O}^*(2.4150^n)$ [17] and an $\mathcal{O}^*(1.4802^n)$ time algorithm for 3-SAT [15]. (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 $f(n)$.) Many natural questions in the field of exact algorithms can not be answered by the classical complexity theory, for example why some NP-hard problems can be solved significantly faster than others? For a good overview of the field see the recent survey written by Gerhard Woeginger [27].

Treewidth is one of the most basic parameters in algorithms and it plays an important role in structural graph theory. It serves as one of the main tools in Robertson & Seymour's Graph Minors project [26]. It is well known that many intractable problems can be solved in polynomial (and very often in linear time) when the input is restricted to graphs of bounded treewidth. In recent years [14] it was shown that the results on graphs of bounded treewidth (branchwidth) are not only of theoretical interest but can successfully be applied to find optimal solutions or lower bounds for diverse optimization problems. (See also [5] for a comprehensive survey.) Treewidth also plays a crucial role in Downey & Fellows parameterized complexity theory [16]. An efficient solution to treewidth is the base for many applications in artificial intelligence, databases and logical-circuit design. To mention just a few of these applications: Exact inference in Bayesian networks, reasoning with structured CSPs (constraint-satisfaction problems), propositional SAT and FOL (first-order logic) problems. See [1] for further references. Finally WWW database applications are practical if they are based on tree decompositions of small width [19].

The Minimum Fill-in problem (also known as minimum chordal graph completion) has important applications in sparse matrix computations and computational biology.

Previous results. Treewidth and minimum fill-in are known to be NP-hard even when the input is restricted to complements of bipartite graphs (so called cobipartite graphs) [2, 28]. Despite of the importance of treewidth almost nothing is known on how to cope with its intractability. It is known that it can be approximated with a factor $\log OPT$ [1, 10] and it is an old open question if the treewidth can be approximated with a constant factor. Treewidth is known to be fixed parameter tractable, moreover, for a fixed k, the treewidth of size k can be computed in linear time (with a huge hidden constant) [4]. There is a number of algorithms that for a given graph G and integer k, either reports that the treewidth of G is at least k, or produces a tree decomposition of width at most $c_1 \cdot k$ in time $O(c_2^k n^{O(1)})$, where c_1, c_2 are some constants (see e.g. [1]). Fixed parameter algorithms are known for fill-in problem as well [12, 22].

There is no previous work on exact algorithms for treewidth or fill-in and almost nothing was known about it. By making use of well known results on the existence of perfect elimination ordering in chordal graphs both problems can be easily reformulated as the problems of finding a special vertex ordering, thus bringing the complexity of both problems to $\mathcal{O}^*(n!)$. This running time can be reduced by a dynamic programming approach computing the treewidth of a graph in time $\mathcal{O}^*(2^n)$ similar to the Held & Karp algorithm [21] for the travelling salesman problem.

Our results. In this paper we break the $\mathcal{O}^*(2^n)$ barrier by obtaining the first exact algorithm for treewidth of running time $\mathcal{O}^*(c^n)$ for $c < 2$. Our algorithm can be adapted not only for treewidth but for a number of other minimal triangulation problems like Minimum Fill-in. Our main result is the $\mathcal{O}^*(1.9601^n)$ 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 [8, 9]. The running time analysis of the algorithm is difficult and is based on a careful counting of potential maximal cliques, i.e. vertex subsets in a graph that can be maximal cliques in some minimal triangulation of this graph. More precisely, we start by modifying the algorithm due to Bouchitté & Todinca [8] to compute the treewdith and minimum fill-in of a graph G with the given set Π_G of all potential maximal cliques of G in time $\mathcal{O}^*(|\Pi_G|)$. Then we obtain a number of structural results which provide us with the proof that $|\Pi_G| = \mathcal{O}^*(1.9601^n)$ and with an algorithm computing all potential maximal cliques in time $\mathcal{O}^*(1.9601^n)$. Our analysis is based on combinatorial bounds for the number of minimal separators as well as for the number of potential maximal cliques in a graph on n vertices. Determining such bounds is an attractive combinatorial problem on its own.

Our algorithms computing treewidth and minimum fill-in need the list of potential maximal cliques and minimal separators at the input. So the running time of the algorithms is bounded by the maximum number of these parameters. For general graphs our combinatorial bounds do not match the lower bounds provided by known examples. However, we prove in Section 6 that the maximum number of minimal separators and potential maximal cliques for the special case of AT-free graphs is $\Theta^*(2^{n/2})$. For these class of graphs our technique can be implemented to obtain algorithms of running time $\mathcal{O}^*(1.4142^n)$. (Note that treewidth and minimum fill-in are NP complete even when restricted to a small subclass of AT-free graphs namely cobipartite graphs.) Thus for this class of graphs we obtained the fastest possible algorithm (up to $\mathcal{O}^*$) that can be obtained by making use of our approach.

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 put $N(S) = \bigcup_{v \in S} N(v) \setminus S$. 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.

The notion of treewidth is due to Robertson & Seymour [25]. 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:

1. $\bigcup_{i \in V_T} X_i = V$;
2. 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
3. for all $v \in V$, the set of nodes $\{i \in V_T | v \in X_i\}$ forms 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 ($\text{tw}(G)$) is the minimum width 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, $\text{tw}(G) \leq k$ if and only if there is a triangulation H of G such that $\omega(H) \leq k + 1$.*

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

The *minimum fill-in* of a graph $G = (V, E)$, denoted by $\text{mfi}(G)$, 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 triangulation with the smallest maximum clique-size, while computing the minimum fill-in means finding a triangulation with the smallest number of edges. In both cases we can restrict our work to minimal triangulations.

Minimal separators and potential maximal cliques. 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 $G[V \setminus S]$. S is a *minimal a, b-separator* 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 Ω 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.

The following result can be used to list all minimal separators in polynomial time.

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

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

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

Theorem 3 ([8]). *Let $K \subseteq V$ be a set of vertices and let $\mathcal{C}(K) = \{C_1(K), \dots, C_p(K)\}$ be the set of the connected components of $G \setminus K$. Furthermore let $\mathcal{S}(K) = \{S_1(K), \dots ,S_p(K)\}$ where $S_i(K)$ is the set of vertices of K adjacent to at least one vertex of $C_i(K)$. Then K is a potential maximal clique if and only if :*

1. *$G \setminus K$ has no full components associated to K, and*
2. *the graph on the vertex set K obtained from $G[K]$ by turning 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.

By Theorem 3, for every potential maximal clique Ω of G, the sets $S_i(\Omega)$ are exactly the minimal separators of G contained in K. For each minimal separator S_i, $\Omega \setminus S_i$ is contained in a component of $G \setminus S_i$.

The following result is an easy consequence of Theorem 3.

Theorem 4 ([8]). *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 [9], 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 $\mathcal{O}^*(1.9601^n)$ potential maximal cliques.

3 Computing treewidth and minimum fill-in

We describe the algorithm of [8] 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. New observation here is that this algorithm can be implemented to run in $\mathcal{O}^*(|\Pi_G|)$ time.

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. 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 5 ([23]). *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* $G[S]$.

Let us note that in the equations of Theorem 5 it is sufficient to take the minimum only over the inclusion-minimal separators of G. In this case all the blocks in the equations are full.

Unfortunately, Theorem 5 is not sufficient for computing the treewidth and the minimum fill-in. We need to 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 6 ([8]). *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 7. *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 $\mathcal{O}^*(|\Pi_G|)$ time. Moreover, the algorithm constructs optimal triangulations for the treewidth and the minimum fill-in.*

Proof's sketch. We provide here only some ideas by moving the detailed proof to the Appendix. W.l.o.g. we can assume that G is connected graph (otherwise we can run the algorithm for each connected component of G). First the algorithm computes all the full blocks and sorts them by their size. If there is no block, then G is a complete graph and in this case the solution is trivial. For each inclusion minimal full block (S, C) (i.e. block containing no other full blocks), we assign $\mathrm{tw}(R(S, C)) = |S \cup C| - 1$ and $\mathrm{mfi}(R(S, C)) = \mathrm{fill}(S \cup C)$. Then, for each full block (S, C) in the increasing order, the treewidth and the minimum fill-in of the realization $R(S, C)$ is computed by making use of Theorem 6. Finally, $\mathrm{tw}(G)$ and $\mathrm{mfi}(G)$ are obtained by Theorem 5.

The time complexity is given by the computation of the treewidth and minimum fill-in parameters for the realizations of all the full blocks, which works as follows:

```
for each full block (S, C)
        for each potential maximal clique Ω such that S ⊂ Ω ⊆ S ∪ C
                apply a polynomial time treatment to each block (S_i, C_i)
                associated to Ω s.t. S_i ∪ C_i ⊂ S ∪ C.
```

When S, C and Ω are fixed, the number of blocks associated to Ω is at most n. Indeed, each of these blocks corresponds to a component of $G \setminus \Omega$. These blocks can be computed in polynomial time. Let us show that the number of triples (S, C, Ω) with $S \subset \Omega \subseteq S \cup C$ is bounded by $n|\Pi_G|$. By Theorem 3, for a fixed potential maximal clique Ω there are at most n minimal separators $S \subset \Omega$. (The number of components in $G \setminus \Omega$, and hence the number of sets $S_i(\Omega)$ is at most n.) Also by Theorem 3, for each minimal separator $S \subset \Omega$, there is a unique component C of $G \setminus S$ such that $\Omega \subseteq S \cup C$.

It remains to turn this argument into an implementation such that the two nested loops have at most $n|\Pi_G|$ iterations. During a preprocessing step, for each potential maximal clique Ω we compute all minimal separators contained in Ω (by Theorem 3, they are the neighborhoods of the components of $G \setminus \Omega$). For each $S \subset \Omega$, we compute the unique component C of $G \setminus S$ containing $\Omega \setminus S$. For each block (S, C) we keep a pointer to Ω. Using appropriate data structures these preprocessing costs $\mathcal{O}(nm)$ for each potential maximal clique. In this way, each block (S, C) will keep the list of all the potential maximal cliques Ω s.t. $S \subset \Omega \subseteq S \cup C$. The number of iterations of the two loops is then the number of “good” triples (S, C, Ω), i.e. at most $n|\Pi_G|$. □

After Theorem 7, the only missing ingredient of our algorithms is listing of all the minimal separators and the potential maximal cliques of a graph in time $\mathcal{O}^*(c^n)$ for some $c < 2$. 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 $\mathcal{O}^*(1.9601^n)$ 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 [20].

Lemma 1 (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.*

Theorem 8. *For any graph G, $|\Delta_G| = \mathcal{O}(n \cdot 1.7087^n)$.*

Let us note, that by Theorem 2, Theorem 8 yields immediately 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 $|\Delta_G| = \#\texttt{small sep} + \#\texttt{big sep}$.

Counting big separators. Let S be a minimal separator. By Lemma 1, 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 $\#\texttt{big sep}$ of big minimal separators is at most the number of small components and we conclude that $\#\texttt{big sep}$ 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:

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

Counting 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. Let (x, X) be a bad pair associated to a minimal separator S. The connected component of $G[X]$ containing x is $C_x(S)$.

Proof. Let C be the component of $G[X]$ containing x. By definition of a bad pair, $C_x(S) \subseteq X$. Since $X \cap S = \emptyset$, the claim follows. $\square$

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

Proof. Suppose that $x = y$ and $X = Y$. By Claim 4, we have $C_x(S) = C_y(T)$. Since $C_x(S)$ is a full component associated to S in G, we have that $S = N(C_x(S))$ and $T = N(C_y(T))$. Thus $S = T$ which is a contradiction. $\square$

By Lemma 1, 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. By Claim 4, 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

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

For $\alpha = 0.5456$, (1) and (2) yield that $\#\texttt{small sep} + \#\texttt{big sep} = \mathcal{O}(n \cdot 1.7087^n)$. $\square$

5 Final step: Counting potential maximal cliques

We prove in this section the main technical result of this paper, namely that a graph with n vertices has at most $\mathcal{O}^*(\binom{n}{2n/5})$ potential maximal cliques, that is $\mathcal{O}^*(1.9601^n)$, enumerable with the same time complexity.

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 $2n/5$. The algorithm for generating all the potential maximal cliques of a graph lists all the sets of vertices of size at most $2n/5$ and then, by applying a polynomial time procedure for each set, generates all the potential maximal cliques of the input graph.

Lemma 2. *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 neighbors 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 3, 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 our 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 $b \in \Omega \setminus S$ 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. □

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. Let us note that for a given triple (S, a, b), one can check in polynomial time whether (S, a, b) is the separator representation of a (unique) potential maximal clique Ω.

The number of all possible separator representations of a graph is bounded by $n^2|\Delta_G|$. Unfortunately, in the case when a potential maximal clique Ω has no separator representation, we cannot say that Ω is small (i.e. of size at most βn for some $\beta < 0.5$) or is the neighborhood of a small set. In the next subsection we introduce another type of representation, the *neighborhood representation*, that allows us to show that all the potential maximal cliques can be represented by small sets of vertices.

Counting nice potential maximal cliques. 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.*

Theorem 9 ([9]). *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 $G[C \cup \{x, y\}]$.*

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. In this subsection we shall prove that a graph with n vertices has $\mathcal{O}^*(\binom{n}{2n/5})$ nice potential maximal cliques.

Lemma 3. *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) 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 9, $\Omega \setminus S$ 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 $D_y = C \setminus D_x$. Notice that the full component associated to $\Omega \setminus S$ in $G[C \cup \{x, y\}]$ and containing y, is in $D_y \cup \{y\}$. Therefore D_x and D_y satisfy the above condition. □

Let us note that if one of the two sets of the partition in Lemma 3, say D_x, is empty, then for any $z \in C$, the triple (S, x, z) is a separator representation of Ω. Let C' be a component of $G \setminus \Omega$ such that $N(C') = S$ and let $c \in C'$. If D_x is not empty, we have that the triple $(X = C' \cup D_x, x, c)$ is sufficient for computing Ω. Indeed, C' is the component of $G[X]$ containing c, $D_x = X \setminus C'$, C is the component of $G - N(C')$ containing D_x, and finally $\Omega = S \cup (\Omega \setminus S) = N(C') \cup (N(D_x \cup \{x\}) \cap C)$. In other words, $(C' \cup D_x, x, c)$ can be used to represent Ω.

More formally: For a potential maximal clique Ω of G, we say that a triple (X, x, c), where $X \subseteq V$, $c \in X$, and $x \notin X$ is a *neighborhood representation* of Ω if the following hold:

1. $D_x = X \setminus C'$ is not empty, where C' is the component of $G[X]$ containing c;

2. $\Omega = N(C') \cup (N(D_x \cup \{x\}) \cap C)$, where C is the component of $G - N(C')$ containing D_x.

Let us remark that for a given triple (X, x, c), we can check in polynomial time whether (X, x, c) is a neighborhood representation of a (unique) potential maximal clique Ω.

We can state now the main tool for counting the nice potential maximal cliques.

Lemma 4. *Let Ω be a nice potential maximal clique of G. Then one of the following holds:*

1. *$|\Omega| \leq 2n/5$;*
2. *There is a vertex a such that $\Omega = N[a]$;*
3. *Ω has a separator representation;*
4. *There is a set of vertices X such that $\Omega = N(X)$ and $|X| \leq 2n/5$;*
5. *Ω has a neighborhood representation (X, x, c) such that $|X| \leq 2n/5$.*

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 3, there is a partition (D_x, D_y) 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 empty, then for any $z \in C$, the triple (S, x, z) is a separator representation of Ω. Suppose that none of the first three lemma's conditions hold. Then D_x and D_y are nonempty. Let C' be a component of $G \setminus \Omega$ such that $N(C') = S$ and $c \in C'$. Since D_x and D_y are not empty, we have that $(C' \cup D_x, x, c)$ and $(C' \cup D_y, y, c)$ are neighborhood representations of Ω.

By Lemma 2, $\Omega = N(C \setminus \Omega)$. Since (D_x, D_y) form a partition of $C \setminus \Omega$, it remains to prove that at least one of the sets $C \setminus \Omega = D_x \cup D_y, C' \cup D_x$ and $C' \cup D_y$ has at most $2n/5$ vertices. Clearly D_x, D_y and C' are pairwise disjoint. Since $|D_x| + |D_y| + |C'| \leq |V(G) \setminus \Omega| \leq 3n/5$, the conclusion follows. ☐

Lemma 5. *A graph on n vertices has at most $n^2 \sum_{i=1}^{2n/5} \binom{n}{i}$ nice potential maximal cliques, enumerable in $\mathcal{O}^*(\binom{n}{2n/5})$ time.*

Proof. The result is a straightforward consequence of Proposition 4. The number of sets of vertices of size at most $2n/5$ is upper bounded by $\sum_{i=1}^{2n/5} \binom{n}{i}$, in particular the number of possible neighborhood representations (X, x, c) with $|X| \leq 2n/5$ is at most $n^2 \sum_{i=1}^{2n/5} \binom{n}{i}$. Notice that by Theorem 8, the number of all possible separator representations is at most $n^2 |\Delta_G| \leq n^2 \binom{n}{2n/5}$ and we deduce that the number of nice potential maximal cliques is at most $n^2 \sum_{i=1}^{2n/5} \binom{n}{i}$. Moreover, these potential maximal cliques can be computed in $\mathcal{O}^*(\binom{n}{2n/5})$ time: 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 $2n/5$ vertices and all the couples of vertices (x, c), we compute all the nice potential maximal cliques of type X, $N(X)$ or having (X, x, c) as a neighborhood representation. ☐

Counting all the potential maximal cliques. Not all the potential maximal cliques of a graph are necessarily nice (see [9] for an example). For counting and enumerating all the potential maximal cliques of a graph, we need the following theorem, used in [9] for showing that the number of potential maximal cliques of G is $\mathcal{O}^*(|\Delta_G|^2)$.

Theorem 10 ([9]). *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 11. *A graph G on n vertices has at most $n^3 \sum_{i=1}^{2n/5} \binom{n}{i} = \mathcal{O}(n^4 \cdot 1.9601^n)$ potential maximal cliques, enumerable in $\mathcal{O}^*(\binom{n}{2n/5}) = \mathcal{O}^*(1.9601^n)$ time.*

Proof. Let $x_1, x_2, \ldots, x_n$ be the vertices of G and $G_i = G[\{x_1, \ldots, x_i\}]$, $\forall i, 1 \le i \le n$. Theorem 10 and Lemma 5 imply that $|\Pi_{G_i}| \le |\Pi_{G_{i-1}}| + n|\Delta_{G_i}| + n^2 \sum_{i=1}^{2n/5} \binom{n}{i}$, for all $i, 2 \le i < n$. By Theorem 8, $|\Pi_G| \le n^3 \sum_{i=1}^{2n/5} \binom{n}{i}$.

Clearly, if we have the potential maximal cliques of G_{i-1}, the potential maximal cliques of G_i can be enumerated in $\mathcal{O}^*(|\Pi_{G_{i-1}}| + \binom{n}{2n/5})$ time by making use of Theorems 10, 8, and Lemma 5. The graph G_1 has a unique potential maximal clique, namely $\{x_1\}$. It follows that Π_G can be computed and enumerated in $\mathcal{O}^*(\binom{n}{2n/5})$ time which is approximately $\mathcal{O}^*(1.9601^n)$. □

Theorems 7 and 11 imply the main result of this paper.

Theorem 12. *For a graph G on n vertices, the treewidth and the minimum fill-in of G can be computed in $\mathcal{O}^*(1.9601^n)$ time.*

6 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 neighbourhood of the third vertex. Graphs without asteroidal triples are called *AT-free*. AT-free graphs contain cocomparability graphs, permutation graphs, interval graphs and cobipartite graphs.

There is a well-known cobipartite graph consisting of two cliques of size $n/2$ and a perfect matching between them which has precisely $2^{n/2} - 2$ minimal separators. Clearly this graph is also AT-free. We want to show that neglecting polynomial factors this is the largest number of minimal separators in an AT-free graph.

Besides a natural graph-theoretic interest in such a question it has also applications in exact treewidth and min fill-in algorithms for AT-free graphs.

In the first part we show that for AT-free graphs we have $|\Pi_G| = \mathcal{O}^*(|\Delta_G|)$, and the potential maximal cliques are enumerable in $\mathcal{O}^*(|\Delta_G|)$ time. Then we prove that an AT-free graph on n vertices has at most $2^{n/2+3}$ minimal separators.

Let us start with some observations on the structure of potential maximal cliques in AT-free graphs.

Lemma 6 ([8]). *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 7 ([8]). *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 $G \setminus S$.*

Consider two minimal separators S and T, incomparable with respect to inclusion. Suppose that T (resp. S) meets exactly one component $C_T(S)$ (resp. $C_S(T)$) of $G \setminus S$ (resp. $G \setminus T$). We define the *piece between* S and T as $P(S,T) = S \cup T \cup (C_T(S) \cap C_S(T))$.

Lemma 8 ([8]). *Let G be a graph and Ω be a potential maximal clique of G such that $\mathcal{S}(\Omega)$ has a two inclusion-maximal elements S and T. Then $\Omega = P(S,T)$.*

Lemma 9. *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 8, $\Omega = P(S,T)$. 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 3, for each minimal separator U contained in Ω, Ω intersects exactly one component of $G \setminus U$, which is a contradiction. □

Theorem 13. *Every AT-free graph G has at most $n^2|\Delta_G| + n|\Delta_G| + 1$ potential maximal cliques, enumerable in $O^*(|\Delta_G|)$ 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 7, 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.

Let us count now the potential 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 9, at most $|S|$ such potential maximal cliques of this type, contained in (S, C). It follows that there are at most n^2 potential maximal cliques of this kind.

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

So by Theorem 7 to find 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 relies on structural properties of a combination of two runs of LexBFS called 2LexBFS (or 2-sweep LexBFS) on AT-free graphs established by Corneil, Olariu & Stewart in [13].

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 neighbours 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 $N_{\downarrow}(v)$.

Two vertices x and y of G form a *dominating pair* if every x,y-path is a dominating set of G.

Theorem 14 ([13]). *Every 2LexBFS ordering $x_n, x_{n-1}, \dots, x_1$ of a connected AT-free graph has the* dominating pair-property *(DP-property), i.e., for all $i \in \{1, 2, \dots, n\}$, (x_n, x_i) is a dominating pair of $G[\{x_i, x_{i+1}, \dots, x_n\}]$.*

The following easy consequence of Theorem 14 has been used before without being explicitely mentioned.

Lemma 10. *Let $x_n, x_{n-1}, \ldots, x_1$ be a 2LexBFS ordering of an AT-free graph G and let $L_0, L_1, \ldots, L_r$ 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 $x_r, w, u_{i-2}, \ldots, u_1, x_n$ with $u_j \in L_j$ and $u_{j-1} \in N^{\uparrow}(u_j)$ for all $j = i-2, \ldots, 1$ contains no neighbour of x_s contradicting the DP-property of any 2LexBFS scheme of an AT-free graph. □

Theorem 15. *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 $x_n, x_{n-1}, \ldots, x_1$ be a 2LexBFS ordering of G and let $L_0, L_1, \ldots, L_r$ 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 $G - S$. We claim that at most one level of the 2LexBFS scheme can contain vertices of C and C'. Suppose not. Let L_i and L_{i+1} be levels containing 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. $u \prec w$. Then Lemma 10 implies that w and v are adjacent, a contradiction.

Let C and C' be two (not necessarily full) components of $G - S$ 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 10 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 neighbour 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 levels. Simply divide the vertex set into two halves: $A = \{x_n, x_{n-1}, \ldots, x_{\lceil n/2 \rceil + 1}\}$ and $B = \{x_{\lfloor n/2 \rfloor}, \ldots, x_1\}$. Then either C or C' is a subset of one 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. Consequently there are at most $2^{n/2+1}$ minimal separators of this type.

It remains to count 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 $G - S$, i.e. $S = N(C) = N(C')$. 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. For similar reason 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 $G' = G - \bigcup_{j=0}^{i-1} L_j$. Then $S' = S \setminus \bigcup_{j=0}^{i-1} L_j$ is a separator of G'; C and C' are components of $G' - S'$. Furthermore, every vertex of $S' \subseteq S$ has a neighbour 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 $f(n) = 4 \cdot 2^{n/2+1} = 8 \cdot 2^{n/2}$. □

Theorems 7, 13, and 15 imply the following improvement of the main result for AT-free graphs.

Theorem 16. *For an AT-free graph G on n vertices, the treewidth and the minimum fill-in of G can be computed in $O^*(1.4142^n)$ time.*

7 Open problems

The running time estimation of our algorithms is based on combinatorial upper bounds on minimal separators and potential maximal cliques. An improvement of these bounds will speed up our algorithms. So an interesting question is whether the upper bounds in Theorems 8 and 11 can be improved. As for the lower bound, we have an example of a graph G 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 $3^{n/3} \sim 1.4422^n$. For some special graph classes, the usage of minimal separators can imply faster algorithms for triangulation problems. For example, we were able to show (we omit the proof in this abstract) that every AT-free graph on n vertices has at most $2^{n/2+3}$ minimal separators. We also can show that this bound is tight up to multiplicative constant factor. Using this bounds we prove that for an AT-free graph G on n vertices, the treewidth and the minimum fill-in of G can be computed in $O^*(1.4142^n)$ time. (Both problems are NP hard on this class of graphs.) The interesting question here is whether similar techniques can be used for bipartite graphs and graphs of small degree.

Our algorithms for treewidth and fill-in can also be applied for solving other problems that can be expressed in terms of minimal triangulations like finding a tree decomposition of minimum cost [7] or computing treewidth of weighted graph.

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 $\mathcal{O}^*(10^n)$ [18] and reducing Feige's bounds is a challenging problem. Pathwidth (and profile) can be expressed as vertex ordering problems and thus solved in $\mathcal{O}^*(n!)$. This time complexity for pathwidth and profile (as for many other vertex ordering problems) can be reduced to $\mathcal{O}^*(2^n)$ by applying a dynamic programming approach similar to Held & Karp's approach [21] 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 highly 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. A. Berry, J.P. Bordat, and O. Cogis, *Generating all the minimal separators of a graph,* In *Workshop on Graph-theoretic Concepts in Computer Science (WG'99)*, volume 1665 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.
4. H. L. Bodlaender, *A linear-time algorithm for finding tree-decompositions of small treewidth*, SIAM J. Comput., 25 (1996), pp. 1305–1317.
5. ———, *A partial k-arboretum of graphs with bounded treewidth*, Theoret. Comput. Sci., 209 (1998), pp. 1–45.
6. 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.
7. H. L. Bodlaender and F. V. Fomin, *Tree decompositions with small cost,* Proceedings of the 8th Scandinavian Workshop on Algorithm Theory (SWAT 2002), M. Penttonen, E. Meineche Schmidt (Eds.): Springer-Verlag Lecture Notes in Computer Science, vol. 2368, 2002, pp. 378–387.
8. V. Bouchitté and I. Todinca, *Treewidth and minimum fill-in: grouping the minimal separators,* SIAM J. on Computing, 31(1):212 – 232, 2001.
9. V. Bouchitté and I. Todinca, *Listing all potential maximal cliques of a graph,* Theoretical Computer Science, 276(1-2):17–32, 2002.
10. V. Bouchitté, D. Kratsch, H. Müller, and I. Todinca, *On treewidth approximation,* Discr. Appl. Math., to appear.
11. H.J. Broersma, E. Dahlhaus, and T. Kloks, *Algorithms for the treewidth and minimum fill-in of HHD-free graphs,* In *Workshop on Graph-theoretic Concepts in Computer Science (WG'97)*, volume 1335 of *Lecture Notes in Computer Science*, pages 109–117. Springer-Verlag, 1997.
12. L. Cai, *Fixed-parameter tractability of graph modification problems for hereditary properties*, Inform. Process. Lett., 58 (1996), pp. 171–176.
13. D.G. Corneil, S. Olariu and L.K. Stewart, *Linear time algorithms for dominating pairs in asteroidal triple-free graphs,* SIAM Journal on Computing 28:1284–1297, 2000.
14. W. Cook and P. Seymour, *Tour merging via branch-decomposition*, INFORMS J. on Computing, 15 (3)(2003), pp. 233–248.
15. E. Dantsin, A. Goerdt, E. A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, and U. Schöning, *A deterministic* $(2-2/(k+1))^n$ *algorithm for k-SAT based on local search*, Theoret. Comput. Sci., 289 (2002), pp. 69–83.
16. R. G. Downey and M. R. Fellows, *Parameterized Complexity*, Springer-Verlag, New York, 1999.
17. D. Eppstein, *Small maximal independent sets and faster exact graph coloring*, in Algorithms and data structures (Providence, RI, 2001), vol. 2125 of Lecture Notes in Comput. Sci., Springer, Berlin, 2001, pp. 462–470.
18. U. Feige, *Coping with the NP-hardness of the graph bandwidth problem*, in Algorithm theory—SWAT 2000 (Bergen), vol. 1851 of Lecture Notes in Comput. Sci., Springer, Berlin, 2000, pp. 10–19.
19. R. Goldman, N. Shivakumar, S. Venkatasubramanian, and H. Garcia-Molina, *Proximity Search in Databases*, VLDB 1998, pp. 26-37
20. M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York, 1980.
21. M. Held and R. Karp, *A dynamic programming approach to sequencing problems*, J. Soc. Indust. Appl. Math., 10 (1962), pp. 196–210.
22. H. Kaplan, R. Shamir, and R. E. Tarjan, *Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs*, SIAM J. Comput., 28 (1999), pp. 1906–1922.
23. T. Kloks, D. Kratsch, and J. Spinrad, *On treewidth and minimum fill-in of asteroidal triple-free graphs,* Theoretical Computer Science, 175:309–335, 1997.
24. A. Parra and P. Scheffler, *Characterizations and algorithmic applications of chordal graph embeddings,* Discrete Appl. Math., 79(1-3):171–188, 1997.
25. N. Robertson and P. D. Seymour, *Graph minors. II. Algorithmic aspects of treewidth*, J. Algorithms, 7 (1986), pp. 309–322.

26. ———, *Graph minors. X. Obstructions to tree-decomposition*, J. Combin. Theory Ser. B, 52 (1991), pp. 153–190.
27. G. J. Woeginger, *Exact algorithms for NP-hard problems: a survey*, in Combinatorial Optimization: "Eureka, you shrink", M. J. et al., ed., Springer, vol. 2570, Berlin, 2003, pp. 185–207.
28. M. Yannakakis, *Computing the minimum fill-in is NP-complete*, SIAM J. Algebraic Discrete Methods, 2 (1981), pp. 77–79.

A Appendix: Proof of Theorem 7

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. A slightly different version of this algorithm is given in [8].

For the sake of completeness and because the algorithm is slightly different, we shortly discuss its proof. According to Theorem 6, 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 5 and the fact that in Theorem 5 one can work only with inclusion-minimal separators.

More precisely, we design the following algorithm:

Input: G, all its potential maximal cliques and all its minimal separators
Output: $\mathrm{tw}(G)$ and $\mathrm{mfi}(G)$
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
    $\mathrm{tw}(R(S,C)) := |S \cup C| - 1$ if (S,C) is inclusion-minimal
      and $\mathrm{tw}(R(S,C)) := \infty$ otherwise
    $\mathrm{mfi}(R(S,C)) := \mathrm{fill}(S \cup C)$ if (S,C) is inclusion-minimal
      and $\mathrm{mfi}(R(S,C)) := \infty$ otherwise
    for each p.m.c. Ω with $S \subset \Omega \subseteq (S,C)$
      compute the blocks (S_i,C_i) associated to Ω s.t. $S_i \cup C_i \subset S \cup C$
      $\mathrm{tw}(R(S,C)) := \min(\mathrm{tw}(R(S,C)), \max_i(|\Omega| - 1, \mathrm{tw}(R(S_i,C_i))))$
      $\mathrm{mfi}(R(S,C)) := \min(\mathrm{mfi}(R(S,C)), \mathrm{fill}(\Omega) - \mathrm{fill}(S) + \sum_i(\mathrm{mfi}(R(S_i,C_i))))$
    end_for
  end_for
  let Δ^*_G be the set of inclusion-minimal separators of G
  $\mathrm{tw}(G) := \min_{S\in\Delta^*_G} \max_{C\in\mathcal{C}(S)} \mathrm{tw}(R(S,C))$
  $\mathrm{mfi}(G) := \min_{S\in\Delta^*_G} (\mathrm{fill}(S) + \sum_{C\in\mathcal{C}(S)} \mathrm{mfi}(R(S,C)))$
end

It remains to discuss an implementation of the algorithm running in $\mathcal{O}^*(|\Pi_G|)$ time.

For keeping the sets of 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, keep 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: compute the components C_i of $G \setminus \Omega$ and the associated blocks are exactly $(N(C_i), C_i)$. 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 keep a pointer from (S, C) to Ω. By Theorem 3, 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 sets, 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 $\mathcal{O}(n^2|\Delta_G| + n^3|\Pi_G|)$. It is not hard to prove (see e.g. [9]) that each minimal separator is contained in at least one potential maximal clique. According to Theorem 3, each potential maximal clique contains at most n minimal separators. Therefore $|\Pi_G| \geq |\Delta_G|/n$. We conclude that the algorithm runs in $\mathcal{O}(n^3|\Pi_G|)$ time.

The algorithm can be easily transformed in order to output not only the treewidth and the minimimum 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 turned into cliques.