

**TREewidth COMPUTATION AND EXTREMAL  
COMBINATORICS\***

FEDOR V. FOMIN, YNGVE VILLANGER

Received September 10, 2008

For a given graph  $G$  and integers  $b, f \geq 0$ , let  $S$  be a subset of vertices of  $G$  of size  $b+1$  such that the subgraph of  $G$  induced by  $S$  is connected and  $S$  can be separated from other vertices of  $G$  by removing  $f$  vertices. We prove that every graph on  $n$  vertices contains at most  $n^{\binom{b+f}{b}}$  such vertex subsets. This result from extremal combinatorics appears to be very useful in the design of several enumeration and exact algorithms. In particular, we use it to provide algorithms that for a given  $n$ -vertex graph  $G$

- compute the treewidth of  $G$  in time  $\mathcal{O}(1.7549^n)$  by making use of exponential space and in time  $\mathcal{O}(2.6151^n)$  and polynomial space;
- decide in time  $\mathcal{O}(n^5 \cdot \binom{\lceil (2n+k+8)/3 \rceil}{k+2})$  if the treewidth of  $G$  is at most  $k$ ;
- list all minimal separators of  $G$  in time  $\mathcal{O}(1.6181^n)$  and all potential maximal cliques of  $G$  in time  $\mathcal{O}(1.7549^n)$ .

This significantly improves previous algorithms for these problems.

**1. Introduction**

The aim of *exact algorithms* is to optimally solve hard problems exponentially faster than brute-force search. The first papers in the area date back to the sixties and seventies [25,36]. For the last two decades the amount of literature devoted to this topic has been tremendous and it is impossible to give here a list of representative references without missing significant results. Recent surveys [18,27,34,38] and book [19] provide a comprehensive information on exact algorithms. It is very natural to assume the existence

---

*Mathematics Subject Classification (2000)*: 05C85, 68R05

\* This research was partially supported by the Research Council of Norway. Extended abstract of this paper was presented in [22].

of strong links between the area of exact algorithms and some areas of extremal combinatorics, especially the part of extremal combinatorics which studies the maximum (minimum) cardinalities of a system of subsets of some set satisfying certain properties. For example, very recently Björklund, Husfeldt, Kaski, Koivisto [4,5] discovered a nice application of Shearer's Lemma in the design of exact algorithms for the travelling salesman problem and coloring on graphs of bounded degrees. Strangely enough, there are not so many examples of such links in the literature, and the majority of exact algorithms are based on the so-called branching (backtracking) technique which traces back to the works of Davis, Putnam, Logemann, and Loveland [14,15].

In this paper, we prove a combinatorial lemma which appears to be very useful in the analysis of certain enumeration and exact algorithms. For a vertex  $v$  of a graph  $G$  and integers  $b, f \geq 0$ , let  $t(b, f)$  be the maximum number of connected induced subgraphs of  $G$  of size  $b+1$  such that the intersection of all these subgraphs is nonempty and each such a subgraph has exactly  $f$  neighbors (a neighbor of a subgraph  $H$  is a vertex of  $G \setminus H$  which is adjacent to a vertex of  $H$ ). Then the combinatorial lemma states that  $t(b, f) \leq \binom{b+f}{b}$  (and it is easy to check that this bound is tight). This can be seen as a variation of Bollobás's Theorem [10], which is one of the corner-stones in extremal set theory. (See Section 9.2.2 of [28] for detailed discussions on Bollobás's Theorem and its variants.)

We use this combinatorial result to obtain faster algorithms for a number of problems related to the treewidth of a graph. The treewidth is a fundamental graph parameter from Graph Minors Theory by Robertson and Seymour [32] and it has numerous algorithmic applications, see the surveys [7,9]. The problems to compute the treewidth is known to be NP-hard [1] and the best known approximation algorithm for treewidth has a factor  $\mathcal{O}(\sqrt{\log OPT})$  [17]. 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 due to Bodlaender [6] computing the treewidth of graphs of treewidth at most  $k$ . Unfortunately, huge hidden constants in the running time of Bodlaender's algorithm is a serious obstacle to its implementation. For small values of  $k$ , the classical algorithm of Arnborg, Corneil and Proskurowski [1] from 1987 which runs in time  $\mathcal{O}(n^{k+3})$  can be used to decide if the treewidth of a graph is at most  $k$ . The first exact algorithm computing the treewidth of an  $n$ -vertex graph is due to Fomin et al. [20] and has running time  $\mathcal{O}(1.9601^n)$ . Later these results were improved in [37] to  $\mathcal{O}(1.8899^n)$ . Both algorithms use exponential space. The fastest polynomial space algo-

rithm for treewidth prior to this work is due to Bodlaender et al. [8] and runs in time  $\mathcal{O}(2.9512^n)$ .

### 1.1. Our results.

We introduce a new (exponential space) algorithm computing the treewidth of a graph  $G$  on  $n$  vertices in time  $\mathcal{O}(1.7549^n)$  and a polynomial space algorithm computing the treewidth in time  $\mathcal{O}(2.6151^n)$ . We also show that if the treewidth of  $G$  is at most  $k$ , then it can be computed in time  $\mathcal{O}(n^5 \cdot \binom{\lceil (2n+k+8)/3 \rceil}{k+2})$ . This is a refinement of the classical result of Arnborg et al. Running times of all these algorithms strongly depend on possibilities of fast enumeration of specific structures in a graph, namely, potential maximal cliques, and minimal separators [8,11,12,21]. The new combinatorial lemma is crucial in obtaining new combinatorial bounds and enumeration algorithms for minimal separators and potential maximal cliques, which, in turn, provides faster algorithms for treewidth.

Similar results can be obtained for a number of parameters related to treewidth. Treelength is a graph parameter introduced by Dourisboure and Gavaille [16] with a motivation from network routing. Graphs of treelength  $k$  are the graphs with tree decompositions such that the distance (in  $G$ ) between any two vertices in the same bag of the decomposition is at most  $k$ . Lokshtanov [30] proved that computing the treelength of a graph is NP-hard and also gave an  $\mathcal{O}(1.8899^n)$  time algorithm solving this problem. The same running time was obtained by Lokshtanov [30] for the Chordal Sandwich problem, where given graphs  $G_1$  and  $G_2$ , one asks if there is a chordal graph  $G$  such that  $G_1 \subseteq G \subseteq G_2$ . As it was observed by Lokshtanov, both problems can be solved in time up to polynomial factor proportional to the number of potential maximal cliques and minimal separators in a graph. Thus our technique allows to solve these problems in time  $\mathcal{O}(1.7549^n)$ . Another application of our results is the minimum fill-in problem (also known as minimum chordal graph completion) which is to turn a given graph into a chordal by adding the minimum number of edges. This problem has important applications in sparse matrix computations [31,33] and database management [2]. The fastest exact algorithm computing the fill-in of a graph known so far runs in time  $\mathcal{O}(1.8899^n)$  [21]. Our techniques can be used to solve the problem in time  $\mathcal{O}(1.7549^n)$ .

The remaining part of the paper is organized as follows. In the next section we provide definitions and preliminary results. In Section 3, we prove our main combinatorial tool. By making use of this tool, in Section 4, we prove combinatorial bounds on the number of minimal separators and po-

tential maximal cliques and obtain algorithm enumerating these structures. These results form the basis for all our algorithms computing the treewidth of a graph presented in Sections 5, 6, and 7.

## 2. Preliminaries

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  $G[W]$ . We say that a vertex set  $S\subseteq V$  is *connected* if  $G[S]$  is connected.

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.

**Minimal separators.** Let  $u$  and  $v$  be two non adjacent vertices of a graph  $G=(V,E)$ . A set of vertices  $S\subseteq V$  is an  $u,v$ -*separator* if  $u$  and  $v$  are in different connected components of the graph  $G[V\setminus S]$ . A connected component  $C$  of  $G[V\setminus S]$  is a *full* component associated to  $S$  if  $N(C)=S$ .  $S$  is a *minimal  $u,v$ -separator* of  $G$  if no proper subset of  $S$  is an  $u,v$ -separator. We say that  $S$  is a *minimal separator* of  $G$  if there are two vertices  $u$  and  $v$  such that  $S$  is a minimal  $u,v$ -separator. Notice that a minimal separator can be strictly included in another one. We denote by  $\Delta_G$  the set of all minimal separators of  $G$ .

We need the following result of Berry et al. [3] (see also Kloks et al. [29]).

**Proposition 2.1 ([3]).** *There is an algorithm listing all minimal separators of an input graph  $G$  in  $\mathcal{O}(n^3|\Delta_G|)$  time.*

The following proposition is an exercise in [23].

**Proposition 2.2 (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  $S$ .*

**Potential maximal cliques.** 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  $E\subseteq E'$ . Triangulation  $H$  is a *minimal triangulation* if for any set  $E''$  with  $E\subseteq E''\subset E'$ , the graph  $F=(V,E'')$  is not chordal.

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  $\Omega$  is a maximal clique of  $H$ . We denote by  $\Pi_G$  the set of all potential maximal cliques of  $G$ .

The following result on the structure of potential maximal cliques is due to Bouchitté and Todinca.

**Proposition 2.3** ([11]). *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  $G[V \setminus K]$  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.  $G[V \setminus K]$  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.

The following result is also due to Bouchitté and Todinca.

**Proposition 2.4** ([11]). *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  $\mathcal{O}(km)$ , where  $k = |K|$ .*

**Proof.** The proof used in [11], states that this verification can be done in  $\mathcal{O}(n^3)$  time. We can make this bound more precise by checking if there are any full components in  $\mathcal{O}(m)$  time, and then check if every vertex of  $K$  can reach any other vertex of  $K$ , by a direct edge or by a path where all intermediate vertices are in  $V \setminus K$ . ■

**Treewidth.** A *tree decomposition* of a graph  $G = (V, E)$  is a pair  $(\chi, T)$  in which  $T = (V_T, E_T)$  is a tree and  $\chi = \{\chi_i | i \in V_T\}$  is a family of subsets of  $V$  such that:

- $\bigcup_{i \in V_T} \chi_i = V$ ;
- for each edge  $e = \{u, v\} \in E$  there exists an  $i \in V_T$  such that both  $u$  and  $v$  belong to  $\chi_i$ ;
- for all  $v \in V$ , the set of nodes  $\{i \in V_T | v \in \chi_i\}$  forms a connected subtree of  $T$ .

To distinguish between vertices of the original graph  $G$  and vertices of  $T$ , we call vertices of  $T$  *nodes* and their corresponding  $\chi_i$ 's *bags*. The maximum size of a bag minus one is called the *width* of the tree decomposition. The *treewidth* of a graph  $G$ ,  $tw(G)$ , is the minimum width over all possible tree decompositions of  $G$ .

An alternative definition of treewidth is via minimal triangulations. The *treewidth* of a graph  $G$  is the minimum of  $\omega(H) - 1$  taken over all triangulations  $H$  of  $G$ . (By  $\omega(H)$  we denote the maximum clique-size of a graph  $H$ .)

Our algorithm for treewidth is based on the following result.

**Proposition 2.5 ([20]).** *There is an algorithm that, given a graph  $G$  together with the list of its minimal separators  $\Delta_G$  and the list of its potential maximal cliques  $\Pi_G$ , computes the treewidth of  $G$  in  $\mathcal{O}(n^3(|\Pi_G| + |\Delta_G|))$  time. Moreover, the algorithm constructs a triangulation  $H$  of  $G$  where the treewidth of  $H$  is the same as the treewidth of  $G$ .*

### 3. Combinatorial Lemma

The following lemma is our main combinatorial tool.

**Lemma 3.1 (Main Lemma).** *Let  $G = (V, E)$  be a graph. For every  $v \in V$ , and  $b, f \geq 0$ , the number of connected vertex subsets  $B \subseteq V$  such that*

- (i)  $v \in B$ ,
- (ii)  $|B| = b + 1$ , and
- (iii)  $|N(B)| = f$

*is at most  $\binom{b+f}{b}$ .*

**Proof.** Let  $v$  be a vertex of a graph  $G = (V, E)$ . For  $b + f = 0$  Lemma trivially holds. We proceed by induction assuming that for some  $k > 0$  and every  $b$  and  $f$  such that  $b + f \leq k - 1$ , Lemma holds. For  $b$  and  $f$  such that  $b + f = k$ , we define  $\mathcal{B}$  as the set of sets  $B$  satisfying (i), (ii), (iii). We claim that

$$|\mathcal{B}| \leq \binom{b+f}{b}.$$

Since the claim always holds for  $b = 0$ , let us assume that  $b > 0$ .

Let  $N(v) = \{v_1, v_2, \dots, v_p\}$ . For  $1 \leq i \leq p$ , we define  $\mathcal{B}_i$  as the set of all connected subsets  $B$  such that

- Vertices  $v, v_i \in B$ ,
- For every  $j < i$ ,  $v_j \notin B$ ,
- $|B| = b + 1$ ,
- $|N(B)| = f$ .

Let us note, that every set  $B$  satisfying the conditions of the lemma is in some set  $\mathcal{B}_i$  for some  $i$ , and that for  $i \neq j$ ,  $\mathcal{B}_i \cap \mathcal{B}_j = \emptyset$ . Therefore,

$$(1) \quad |\mathcal{B}| = \sum_{i=1}^p |\mathcal{B}_i|.$$

For every  $i > f + b - 1$ ,  $|\mathcal{B}_i| = 0$ . This is because for every  $B \in \mathcal{B}_i$ , the set  $N(B)$  contains vertices  $v_1, \dots, v_{i-1}$  and thus is of size at least  $f + b > f$ . Thus, (1) can be rewritten as follows

$$(2) \quad |\mathcal{B}| = \sum_{i=1}^{f+b-1} |\mathcal{B}_i|.$$

Let  $G_i$  be the graph obtained from  $G$  by contracting edge  $\{v, v_i\}$  (removing the loop, reducing double edges to single edges, and calling the new vertex by  $v$ ) and removing vertices  $v_1, \dots, v_{i-1}$ . Then the cardinality of  $\mathcal{B}_i$  is equal to the number of the connected vertex subsets  $B$  of  $G_i$  such that

- $v \in B$ ,
- $|B| = b$ ,
- $|N(B)| = f - i + 1$ .

By the induction assumption, this number is at most  $\binom{f+b-i}{b-1}$ . Then (2) and the well known identity

$$(3) \quad \sum_{i=0}^p \binom{p-i}{k} = \binom{p+1}{k+1}$$

with  $p = f + b - 1$  and  $k = b - 1$ , yield that

$$\begin{aligned} |\mathcal{B}| &= \sum_{i=1}^{f+b-1} |\mathcal{B}_i| \leq \sum_{i=1}^{f+b-1} \binom{f+b-i}{b-1} \\ &\leq \sum_{i=0}^{f+b-1} \binom{f+b-1-i}{b-1} = \binom{b+f}{b}. \quad \blacksquare \end{aligned}$$

The inductive proof of Main Lemma can be easily turned into a recursive polynomial space enumeration algorithm. Equation (2) corresponds to a recursive or branching procedure that is called for each  $i \leq f + b - 1$  on graphs obtained from the original graph by deleting the first  $i$  neighbors of  $v$  and contracting edge between  $v$  and its  $(i + 1)$ th neighbor. The proof of Main Lemma yields that the number of leaves in the branching tree is at most  $\binom{b+f}{b}$ , and as  $b$  is decreased with each call, there are at most  $b \binom{b+f}{b}$  recursive calls.

Because the degree of  $v$  is at most  $f + b$ , with the corresponding data structure, each recursive call can be implemented to run in time  $\mathcal{O}(n(b+f))$ . All the sets can now be enumerated in polynomial space by starting a search from each vertex of the graph using a DFS exploration of the branching tree.

We obtain the following lemma. We run the recursive procedure for every vertex of a graph, so in total the running time will be  $\mathcal{O}(n^2 \cdot b \cdot (b+f) \cdot \binom{b+f}{b})$ .

**Lemma 3.2.** *All connected vertex sets of size  $b+1$  with  $f$  neighbors of an  $n$ -vertex graph  $G$  can be enumerated in time  $\mathcal{O}(n^2 \cdot b \cdot (b+f) \cdot \binom{b+f}{b})$  by making use of polynomial space.*

### 4. Combinatorial bounds

In this section we provide combinatorial bounds on the number of minimal separators and potential maximal cliques in a graph. Both bounds are obtained by making use of Main Lemma.

#### 4.1. Minimal separators

**Theorem 4.1.** *Let  $\Delta_G$  be the set of all minimal separators in a graph  $G$  on  $n$  vertices. Then  $|\Delta_G| = \mathcal{O}(1.6181^n)$ .*

**Proof.** For  $1 \leq i \leq n$ , let  $s(i)$  be the number of all minimal separators in  $G$  of size  $i$ . Then

$$(4) \quad |\Delta_G| = \sum_{i=1}^n s(i).$$

Let  $S$  be a minimal separator of size  $\lfloor \alpha n \rfloor$ , where  $0 < \alpha < 1$ . By Proposition 2.2, there exists two full components  $C_1$  and  $C_2$  associated to  $S$ . Let us assume that  $|C_1| \leq |C_2|$ . Then  $|C_1| \leq \lfloor (1-\alpha)n/2 \rfloor$ . From the definition of a full component  $C_1$  associated to  $S$ , we have that  $N(C_1) = S$ . Thus,  $s(\lfloor \alpha n \rfloor)$  is at most the number of connected vertex sets  $C$  of size at most  $\lfloor (1-\alpha)n/2 \rfloor$  with neighborhoods of size  $|N(C)| = \lfloor \alpha n \rfloor$ . Hence, to bound  $s(\lfloor \alpha n \rfloor)$  we can use Main Lemma for every vertex of  $G$ .

By Main Lemma, we have that for every vertex  $v$ , the number of full components of size  $b+1 = \lfloor (1-\alpha)n/2 \rfloor$  containing  $v$  and with neighborhoods of size  $\lfloor \alpha n \rfloor$  is at most

$$\binom{b + \lfloor \alpha n \rfloor}{b} \leq \binom{\lfloor (1 + \alpha)n/2 \rfloor}{b}.$$

Therefore

$$(5) \quad s(\lfloor \alpha n \rfloor) \leq n \cdot \sum_{i=1}^{\lfloor (1-\alpha)n/2 \rfloor} \binom{i + \lfloor \alpha n \rfloor}{i} < n \cdot \sum_{i=1}^{\lfloor (1-\alpha)n/2 \rfloor} \binom{\lfloor (1 + \alpha)n/2 \rfloor}{i}.$$

For  $\alpha \leq 1/3$ , we have

$$\sum_{i=1}^{\lfloor (1-\alpha)n/2 \rfloor} \binom{\lfloor (1+\alpha)n/2 \rfloor}{i} < 2^{\lfloor (1+\alpha)n/2 \rfloor} < 2^{\lfloor 2n/3 \rfloor} < 1.59^n,$$

and thus

$$(6) \quad \sum_{i=1}^{\lfloor n/3 \rfloor} s(i) = \mathcal{O}(1.59^n).$$

For  $\alpha \geq 1/3$ , we use the well known fact, see e.g. [24] or [19, Lemma 3.23] that

$$\sum_{k=0}^{\lfloor j/2 \rfloor} \binom{j-k}{k} = F(j+1),$$

where

$$F(j+1) = \left\lfloor \frac{\varphi^{j+1}}{\sqrt{5}} + \frac{1}{2} \right\rfloor$$

is the  $(j+1)$ -st Fibonacci number and  $\varphi = (1 + \sqrt{5})/2$  is the Golden Ratio. Then

$$\begin{aligned} \sum_{i=1}^{\lfloor (1-\alpha)n/2 \rfloor} \binom{\lfloor (1+\alpha)n/2 \rfloor}{i} &\leq \sum_{i=1}^{\lfloor (1-\alpha)n/2 \rfloor} \binom{n-i}{i} \\ &\leq \sum_{i=1}^{\lfloor n/2 \rfloor} \binom{n-i}{i} < \varphi^{n+1} < 1.6181^n. \end{aligned}$$

Therefore,

$$(7) \quad \sum_{i=\lfloor n/3 \rfloor}^n s(i) = \mathcal{O}(1.6181^n).$$

Finally, the theorem follows from (4),(6) and (7). ■

### 4.2. Potential maximal cliques

A clique contains exactly one potential maximal clique, so for the rest of this paper we will consider the problem of listing potential maximal cliques on non-complete graphs. To estimate the number of potential maximal cliques, we start with establishing bounds on specific types of potential maximal cliques.

**Definition 4.2** ([11]). Let  $\Omega$  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  $\Omega$ , if  $\Omega$  is not a clique in the graph obtained from  $G$  by completing all the minimal separators contained in  $\Omega$ , except  $S$ . A potential maximal clique  $\Omega$  containing an active separator (for  $\Omega$ ) is called a *nice potential maximal clique*.

We need the following result by Bouchitté and Todinca.

**Proposition 4.3** ([12]). *Let  $\Omega$  be a potential maximal clique of  $G = (V, E)$ , let  $u$  be a vertex of  $G$ , and let  $G' = G[V \setminus \{u\}]$ . Then one of the following holds:*

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

Let  $\Pi_n$  be the maximum number of nice potential maximal cliques that can be contained in a graph on  $n$  vertices. Proposition 4.3 is useful to bound the number of potential maximal cliques in a graph by the number of minimal separators  $\Delta_G$  and  $\Pi_n$ . The following was proved in [21].

**Proposition 4.4** ([21]). *For any graph  $G = (V, E)$ ,  $|\Pi_G| \leq n(n|\Delta_G| + \Pi_n)$ .*

**Definition 4.5.** Let  $\Omega \in \Pi_G$ ,  $v \in \Omega$ , and  $C_v$  be the connected component of  $G[V \setminus (\Omega \setminus \{v\})]$  containing  $v$ . We call the pair  $(C_v, v)$  by *vertex representation* of  $\Omega$ .

The following proposition was proved in [21].

**Proposition 4.6** ([21]). *Let  $(C_v, v)$  be a vertex representation of  $\Omega$ . Then  $\Omega = N(C_v) \cup \{v\}$ .*

**Proof.** By Proposition 2.3, every vertex  $u \in \Omega \setminus \{v\}$ , is either adjacent to  $v$ , or there exists a connected component  $C$  of  $G[V \setminus \Omega]$  such that  $u, v \in N(C)$ . Since  $C \subset C_v$ , we have that  $\Omega \setminus \{v\} \subseteq N(C_v)$ . Every connected component  $C$  of  $G[V \setminus \Omega]$  that contains  $v \in N(C)$  is contained in  $C_v$  and  $N(C) \subset \Omega$  for every  $C$ , therefore  $\Omega \setminus \{v\} = N(C_v)$ . ■

We need also the following result from [21].

**Proposition 4.7** ([21]). *Let  $\Omega$  be a nice potential maximal clique of size  $\lfloor \alpha n \rfloor$  for  $0 < \alpha < 1$  in a graph  $G$ . There exists a vertex representation  $(C_v, v)$  of  $\Omega$  such that*

$$|C_v| \leq \left\lceil \frac{2(1 - \alpha)n}{3} \right\rceil.$$

Now everything is settled to apply Main Lemma.

**Lemma 4.8.** *The number of nice potential maximal cliques in graph  $G$  is  $\mathcal{O}(1.7549^n)$ .*

**Proof.** By Proposition 4.7, for every nice potential maximal clique  $\Omega$  of cardinality  $\lfloor \alpha n \rfloor$ ,  $0 < \alpha \leq 1$ , there exists a vertex representation  $(C_v, v)$  of  $\Omega$  such that  $|C_v| \leq \lceil 2n(1-\alpha)/3 \rceil$ . Let  $b+1$  be the number of vertices in  $C_v$  and  $f$  be  $|N(C_v)| = |\Omega \setminus \{v\}| \leq \lfloor \alpha n \rfloor - 1$ . By Main Lemma, for every vertex  $v$ , the number of such pairs  $(C_v, v)$  is at most

$$\sum_{i=1}^{\lceil \frac{2(1-\alpha)n}{3} \rceil} \binom{\lceil (2+\alpha)n/3 \rceil}{i}.$$

As in the proof of Theorem 4.1, for  $\alpha \leq 2/5$  the above sum is  $\mathcal{O}(1.7549^n)$ . For  $\alpha \geq 2/5$ , by making use of the fact that  $\sum_{k=1}^{\lfloor j/2 \rfloor} \binom{j-k}{2k}$  is equal to the  $(j+1)$ -st number of the sequence  $\{a_i\}_{i=0}^\infty$  such that  $a_i = 2a_{i-1} - a_{i-2} + a_{i-3}$ , with  $a_0 = 0$ ,  $a_1 = 1$ , and  $a_2 = 2$ , it is possible to show that the value of the above sum, and thus the number of nice potential maximal cliques, is  $\mathcal{O}(1.7549^n)$ . ■

By combining Proposition 4.4, Lemma 4.8 and Theorem 4.1 we arrive at the main result of this subsection.

**Theorem 4.9.** *For any graph  $G$ ,  $|II_G| = \mathcal{O}(1.7549^n)$ .*

### 5. Exact algorithm for treewidth

Our algorithm computing the treewidth of a graph is based on Proposition 2.5. In order to use Proposition 2.5, we need algorithms enumerating minimal separators and potential maximal cliques. By Proposition 2.1 and Theorem 4.1, all minimal separators can be listed in time  $\mathcal{O}(1.6181^n)$ .

In the following lemma we show how to list all potential maximal cliques.

**Lemma 5.1.** *For any graph  $G$  on  $n$  vertices, the set of potential maximal cliques can be listed in  $\mathcal{O}(1.7549^n)$  time.*

**Proof.** The algorithm listing potential maximal cliques has two phases. First it generates all nice potential maximal cliques (and here again we use Main Lemma). In the second phase by making use of Proposition 4.3, we generate potential maximal cliques from minimal separators and the nice ones.

*Generating nice potential maximal cliques.* For every vertex  $v$  and  $0 < \alpha < 1$  we generate all nice potential maximal cliques of size  $\lfloor \alpha n \rfloor$  containing vertex  $v$ . By Proposition 4.6, to generate all such potential maximal cliques it is sufficient to generate all their vertex representations. By Proposition 4.7, we can restrict our search to vertex representations of size at most  $\lceil 2n(1-\alpha)/3 \rceil$ . For every vertex representation  $C$ , we have that  $|N(C) \cup \{v\}| = \lfloor \alpha n \rfloor$  (the set  $N(C) \cup \{v\}$  should form a potential maximal clique). For each such subset we use Proposition 2.4 to check if  $N(C) \cup \{v\}$  is a potential maximal clique.

So finally, the problem of generating all nice potential maximal cliques boils down to the following problem: List all connected sets  $B$  such that

$$|B| \leq \left\lceil \frac{2n(1-\alpha)}{3} \right\rceil \text{ and } |N(B) \cup \{v\}| \leq \lfloor \alpha n \rfloor.$$

By plugging into algorithmic version of Main Lemma (Lemma 3.2)  $b \leq \lceil 2n(1-\alpha)/3 \rceil - 1$  and  $f = \lfloor \alpha n \rfloor$ , we obtain that for each  $0 < \alpha < 1$  all nice potential maximal cliques of size  $\lfloor \alpha n \rfloor$  can be listed in time

$$(8) \quad \mathcal{O}\left(n^4 \sum_{i=1}^{\lceil \frac{2n(1-\alpha)}{3} \rceil} \binom{\lceil (2+\alpha)n/3 \rceil}{i}\right)$$

By using Main Lemma for each value of  $\alpha$  (where  $\alpha n = \lfloor \alpha n \rfloor$ ) and by bounding the sum as in the proof of Lemma 4.8, we end up with an algorithm that lists all nice potential maximal cliques in time  $\mathcal{O}(1.7549^n)$ .

*Generating potential maximal cliques.* First we generate all minimal separators and all potential maximal cliques that can be obtained from a minimal separator by adding one vertex. Then we list all nice potential maximal cliques. As we already have shown, all these sets can be listed in time  $\mathcal{O}(1.7549^n)$ .

Let  $v_1, v_2, \dots, v_n$  be an ordering of  $V$  and let  $V_i = \bigcup_{j=1}^i v_j$ . By Proposition 4.3, every potential maximal clique  $\Omega$  in  $G[V_{i+1}]$  that is not yet listed (i.e. is not obtained by adding a vertex to a minimal separator or which is not nice) is either also a potential maximal clique in  $G[V_i]$  or is obtained from a potential maximal clique in  $G[V_i]$  by adding  $v_{i+1}$ . Thus by performing dynamic programming one can list all remaining potential maximal cliques in time  $\mathcal{O}(1.7549^n)$ . ■

As an immediate corollary of Proposition 2.1 and Lemma 5.1, we have the following result.

**Theorem 5.2.** *The treewidth of a graph on  $n$  vertices can be computed in time  $\mathcal{O}(1.7549^n)$ .*

### 6. Computing treewidth at most $k$

In this section we show how the lemma bounding the number of connected components can be used to refine the classical result of Arnborg et al. [1].

By Proposition 2.5, the treewidth of a graph can be computed in  $\mathcal{O}(n^3 \cdot (|\Pi_G| + |\Delta_G|))$  time if the list of all minimal separators  $\Delta_G$  and the list of all potential maximal cliques  $\Pi_G$  of  $G$  are given. Actually, the results of Proposition 2.5 can be refined (with exactly the same proof as in [21]) as follows. Let  $\Delta_G^k$  be the set of minimal separators and let  $\Pi_G^k$  be the set of potential maximal cliques of size at most  $k$ .

**Lemma 6.1.** *Given a graph  $G$  with sets  $\Delta_G^k$  and  $\Pi_G^{k+1}$ , it can be decided in time  $\mathcal{O}(n^3 (|\Pi_G^{k+1}| + |\Delta_G^k|))$  if the treewidth of  $G$  is at most  $k$ . Moreover, if the treewidth of  $G$  is at most  $k$ , a tree decomposition of minimum width can be constructed within the same time.*

We first bound the number of minimal separators of size  $k$  in an  $n$ -vertex graph.

**Lemma 6.2.** *For every graph  $G$  on  $n$  vertices,*

$$|\Delta_G^k| \leq n \cdot \binom{\lfloor (n+k+4)/2 \rfloor}{k+1}.$$

**Proof.** By Main Lemma, (5), and (3), we have

$$\begin{aligned} |\Delta_G^k| &\leq n \cdot \sum_{i=1}^{\lfloor (n-k)/2 \rfloor} \sum_{j=1}^k \binom{i+j}{i} \\ &\leq n \cdot \sum_{i=1}^{\lfloor (n-k)/2 \rfloor} \binom{i+k+1}{i+1} = n \cdot \sum_{i=1}^{\lfloor (n-k)/2 \rfloor} \binom{i+k+1}{k} \\ (9) \quad &\leq n \cdot \binom{\lfloor (n+k+4)/2 \rfloor}{k+1}. \quad \blacksquare \end{aligned}$$

For each connected vertex set with at most  $k$  neighbors, with a help of Proposition 2.2, we can check in time  $\mathcal{O}(m)$  if it is a minimal separator or not, here  $m$  is the number of edges in the input graph. In addition to this we can use the fact that  $m \leq kn$  for a graph of treewidth  $k$ . Thus by Lemma 3.2, all minimal separators of size  $k$  can be listed in time  $\mathcal{O}(n^4 \cdot \binom{\lfloor (n+k+4)/2 \rfloor}{k+1})$ .

Let  $\Pi_n^k$  be the maximum number of nice potential maximal cliques of size at most  $k$  that can be in a graph on  $n$  vertices.

By Proposition 4.7,

$$\begin{aligned}
 \Pi_n^k &\leq n \cdot \sum_{i=1}^{\lceil (n-k)2/3 \rceil} \sum_{j=1}^k \binom{i+j}{i} \\
 &\leq n \cdot \sum_{i=1}^{\lceil (n-k)2/3 \rceil} \binom{i+k+1}{i+1} = n \cdot \sum_{i=1}^{\lceil (n-k)2/3 \rceil} \binom{i+k+1}{k} \\
 (10) \quad &\leq n \cdot \binom{\lceil (2n+k+6)/3 \rceil}{k+1}.
 \end{aligned}$$

Then by Lemma 3.2 and Proposition 2.4 all nice potential maximal cliques of size at most  $k$  can be listed in time  $\mathcal{O}(n^4 \cdot \binom{\lceil (2n+k+6)/3 \rceil}{k+1})$ .

Finally, we use nice potential maximal cliques and minimal separators of size  $k$  to generate all potential maximal cliques of size at most  $k$ .

**Lemma 6.3.** *For every graph  $G$  on  $n$  vertices,  $|\Pi_G^k| \leq n(|\Delta_G^k| + \Pi_n^k)$  and all potential maximal cliques of  $G$  of size at most  $k$  can be listed in time  $\mathcal{O}(n^5 \cdot \binom{\lceil (2n+k+6)/3 \rceil}{k+1})$ .*

**Proof.** Let  $v_1, v_2, \dots, v_n$  be an ordering of  $V$  and let  $V_i = \bigcup_{j=1}^i v_j$ . By Propositions 4.3 and 4.4, every potential maximal clique of  $G[V_i]$  is either a nice potential maximal clique of  $G[V_i]$ , or a potential maximal clique of  $G[V_{i-1}]$ , or is obtained by adding  $v_i$  to a minimal separator or a potential maximal clique of  $G[V_{i-1}]$ . This yields that  $|\Pi_G^k| \leq n(|\Delta_G^k| + \Pi_n^k)$ . To list all potential maximal cliques, for each  $i$ ,  $1 \leq i \leq n$ , we list all minimal separators and nice potential maximal cliques in  $G[V_i]$ . This can be done in time  $\mathcal{O}(n^4 \cdot \binom{\lceil (2n+k+6)/3 \rceil}{k+1})$ . The total number of all such structures is at most  $2n \cdot \binom{\lceil (2n+k+6)/3 \rceil}{k+1}$ . By making use of dynamic programming, one can check if adding  $v_i$  to a minimal separator or potential maximal clique of  $G[V_{i-1}]$  creates a potential maximal clique in  $G[V_i]$ , which by Proposition 2.4 can be done in time  $\mathcal{O}(km)$ . Thus, dynamic programming can be done in  $\mathcal{O}(n^5 \cdot \binom{\lceil (2n+k+6)/3 \rceil}{k+1})$  steps, since  $m \leq kn$  if the treewidth is at most  $k$ . ■

Now putting Lemmata 6.1, 6.3, and Equation (9) together, we obtain the main result of this section.

**Theorem 6.4.** *There exists an algorithm that for a given graph  $G$  and integer  $k \geq 0$ , either computes a tree decomposition of  $G$  of the minimum width, or correctly concludes that the treewidth of  $G$  is at least  $k + 1$ . The running time of this algorithm is  $\mathcal{O}(n^5 \cdot \binom{\lceil (2n+k+8)/3 \rceil}{k+2})$ .*

**Proof.** By Lemma 6.3, graph  $G$  contains at most  $2n^2 \cdot \binom{\lceil (2n+k+8)/3 \rceil}{k+2}$  minimal separators and potential maximal cliques of size at most  $k+1$  and these can be enumerated in  $\mathcal{O}(n^5 \cdot \binom{\lceil (2n+k+8)/3 \rceil}{k+2})$  time. Now the proof follows from Lemma 6.1. ■

### 7. Polynomial space algorithm for treewidth

The algorithm used in Proposition 2.1 requires exponential space because it is based on dynamic programming which keeps a table with all potential maximal cliques. As a consequence our  $\mathcal{O}(1.7549^n)$  time algorithm for computing the treewidth also uses  $\mathcal{O}(1.7549^n)$  space.

When restricting to polynomial space, we cannot store all the minimal separators and all the potential maximal cliques. The idea used to avoid this is to search for a “central” potential maximal clique or a minimal separator in the graph which can safely be completed into a clique. A similar idea is used in [8], however, the improvement in the running time of our algorithm, is due to the following lemma and the technique used for listing minimal separators. Both results are, again, based on Main Lemma.

**Lemma 7.1.** *For a given graph  $G = (V, E)$  and  $0 < \alpha < 1$ , one can list in time  $\mathcal{O}(n^4 \cdot 2^{n(1-\alpha)})$  and polynomial space all potential maximal cliques of  $G$  such that for every potential maximal clique  $\Omega$  from this list, there is a connected component of  $G[V \setminus \Omega]$  of size at least  $\alpha n$ .*

**Proof.** Let  $\Omega$  be a potential maximal clique satisfying the conditions of the lemma, and let  $C$  be the connected component of size at least  $\lfloor \alpha n \rfloor$ . By Proposition 2.3,  $N(C)$  is a minimal separator contained in  $\Omega$  and  $\Omega \setminus N(C) \neq \emptyset$ . Let  $(C_u, u)$  be a vertex representation of  $\Omega$ , where  $u \in \Omega \setminus N(C)$ . Since no vertex of  $C$  is adjacent to  $u$ , we have that  $C_u \cap C = \emptyset$ . To find  $\Omega$ , we try to find its vertex representation by a connected vertex set such that the closed neighborhood of this set is of size at most  $\lfloor n(1-\alpha) \rfloor$ . By Main Lemma, the number of such sets is at most

$$n \cdot \sum_{i=1}^{\lfloor n(1-\alpha) \rfloor} \binom{\lfloor n(1-\alpha) \rfloor}{i} \leq n \cdot 2^{n(1-\alpha)},$$

and by Lemma 3.2, all these sets can be listed in  $\mathcal{O}(n^4 \cdot 2^{n(1-\alpha)})$  steps and within polynomial space. Finally, for each set we use Propositions 4.6 and 2.4 to check in time  $\mathcal{O}(mn)$  if the set is a potential maximal clique. ■

We also use the following result which is a slight modification of the result from [8], where it is stated in terms of elimination orderings.

**Proposition 7.2 ([8]).** *For a given graph  $G = (V, E)$  and a clique  $K \subset V$ , there exists a polynomial space algorithm, that computes the optimum tree decomposition  $(\chi, T)$  of  $G$ , subject to the condition that the vertices of  $K$  form a bag which is a leaf of  $T$ . This algorithm runs in time  $\mathcal{O}^*(4^{n-|K|})$ .*

**Theorem 7.3.** *The treewidth of a graph  $G = (V, E)$  can be computed in  $\mathcal{O}(2.6151^n)$  time and polynomial space.*

**Proof.** It is well known (and follows from the properties of clique trees of chordal graphs), that there is an optimal tree decomposition  $(\chi, T)$ ,  $\{\chi_i : i \in V_T\}$ ,  $T = (V_T, E_T)$ , of  $G$ , where every bag is a potential maximal clique [11,13,26]. Among all the bags of  $\chi$ , let  $\chi_i$  be a bag such that the largest connected component of  $G[V \setminus \chi_i]$  is of minimum size, i.e.  $\chi_i$  is a bag with the minimum value of

$$\max\{ |C| : C \text{ is a connected component of } G[V \setminus \chi_i] \},$$

where minimum is taken over all bags of  $\chi$ . Let  $C_i$  be the connected component of  $G - \chi_i$  of maximum size.

Our further strategy depends on the size of  $|C_i|$ . Let us assume first that  $|C_i| < 0.38685n$ . In this case, by Lemma 7.1, the set of potential maximal cliques  $\mathcal{S}$  such that for every  $\Omega \in \mathcal{S}$  the maximum size of a component of  $G[V \setminus \Omega]$  is  $|C_i|$ , can be listed in time  $\mathcal{O}(n^4 \cdot 2^{n-|C_i|})$  and polynomial space. Since  $\chi_i \in \mathcal{S}$ , we have that there is a potential maximal clique  $\Omega \in \mathcal{S}$  such that  $tw(G_\Omega) = tw(G)$ , where  $G_\Omega$  is obtained from  $G$  by turning  $\Omega$  into a clique. The treewidth of  $G_\Omega$  is equal to the maximum of minimum width of decompositions of  $G_\Omega[C \cup \Omega]$  with  $\Omega$  forming a leaf bag, where  $C$  is a connected component of  $G_\Omega[V \setminus \Omega]$ . Let us remind that the size of each such component is at most  $|C_i|$ .

By Proposition 7.2, the optimum width of  $G_\Omega[C \cup \Omega]$  for every connected component  $C$  of  $G_\Omega[V \setminus \Omega]$  (and with  $\Omega$  forming a leaf bag) can be computed in  $\mathcal{O}^*(4^{|C|}) = \mathcal{O}^*(4^{|C_i|})$ , time and thus the treewidth of  $G$  can be found in time

$$\mathcal{O}^*(2^{n-|C_i|} \cdot 4^{|C_i|}) = \mathcal{O}^*(2^{(1-0.38685)n} \cdot 4^{0.38685n}) = \mathcal{O}(2.6151^n).$$

Thus if  $|C_i| < 0.38685n$ , we compute the treewidth of  $G$ , and the running time of this polynomial space procedure is  $\mathcal{O}(2.6151^n)$ .

Let us consider the case  $|C_i| \geq 0.38685n$ . For each connected component  $C$  of  $G[V \setminus \chi_i]$ , there exists a bag  $\chi_{i'} \subset N(C) \cup C$  and a minimal separator  $S = \chi_i \cap \chi_{i'}$  in  $\chi_i$  that separates  $C$  from the rest of the graph. Let  $S = \chi_i \cap \chi_j$  be the separator in  $\chi_i$  that separates  $C_i$  from the rest of the graph. Let  $G_S$

be the graph obtained from  $G$  by turning  $S$  into a clique. Then  $tw(G_S) = tw(G)$ . To compute the treewidth of  $G_S$  we compute the minimum width of decompositions of  $G_S[C \cup S]$  with  $S$  forming a leaf bag, where  $C$  is a connected component of  $G_S[V \setminus S]$ , and then take the maximum of these values.

By the definition of  $\chi_i$ , there exists a connected component  $C_j$  of  $G[V \setminus \chi_j]$ , such that  $|C_j| \geq |C_i|$ . By Proposition 2.3,  $\chi_j \not\subseteq \chi_i$ . Thus  $\chi_j \setminus \chi_i \neq \emptyset$ , and the size of every connected component in  $G[C_i \setminus \chi_j]$  is at most  $|C_i| - 1$ . Furthermore, since  $S = \chi_i \cap \chi_j$ , we have that every connected component of  $G[C_i \setminus \chi_j]$  is also a connected component of  $G[V \setminus \chi_j]$ . This yields that  $C_j \cap C_i = \emptyset$  and that both  $C_i$  and  $C_j$  are full connected components associated to  $S$ . Thus  $|C_j| + |C_i| \leq n - |S|$ . Every connected component of  $G[V \setminus S]$ , except  $C_i$ , is a connected component of  $G[V \setminus \chi_j]$ . Because  $|C_i| \leq |C_j|$ , this implies that  $C_j$  is the largest component of  $G[V \setminus S]$ . Both  $C_i$  and  $C_j$  contain at least  $0.38685n$  vertices, thus the size of  $S$  is at most  $n(1 - 2 \cdot 0.38685) = 0.2263n$ . By the algorithmic version of Main Lemma, all sets of such size (and which form the neighborhood of a set of size  $|C_i|$ ) can be listed in polynomial space and time

$$\mathcal{O}(n^4 \cdot \sum_{p=1}^{0.2263n} \binom{|C_i| + p}{p}).$$

By Proposition 7.2, we can compute the minimum width of decompositions of  $G_S[C \cup S]$  with  $S$  forming a leaf bag, where  $C$  is a connected component of  $G_S[V \setminus S]$ , in time

$$\mathcal{O}^*(4^{|C|}) = \mathcal{O}^*(4^{|C_j|})$$

and polynomial space. Because  $|C_j| \leq n - |S| - |C_i|$ , we have that for  $|S| = p$ ,

$$\mathcal{O}^*(4^{|C_j|}) = \mathcal{O}^*(4^{n - |C_i| - p}).$$

Thus to compute the treewidth of  $G_S$  (and the treewidth of  $G$ ), we list all sets  $S$  and for each such set we use Proposition 7.2 for all graphs  $G_S[C \cup S]$ . The running time of this procedure is

$$\mathcal{O}^* \left( \sum_{p=1}^{0.2263n} \binom{|C_i| + p}{p} \cdot 4^{n - |C_i| - p} \right).$$

By Vandermonde's identity, we have that

$$\begin{aligned} \binom{|C_i| + p}{p} &= \sum_{k=0}^p \binom{0.38685n + p}{k} \binom{|C_i| - 0.38685n}{p - k} \\ &< \sum_{k=0}^p \binom{0.38685n + p}{k} 2^{|C_i| - 0.38685n}. \end{aligned}$$

Thus

$$\begin{aligned} \sum_{p=1}^{0.2263n} \binom{|C_i| + p}{p} 4^{n - |C_i| - p} &< \sum_{p=1}^{0.2263n} \sum_{k=0}^p \binom{0.38685n + p}{k} 2^{|C_i| - 0.38685n} 4^{n - |C_i| - p} \\ &\leq \sum_{p=1}^{0.2263n} p \binom{0.38685n + p}{p} \cdot 2^{2((1 - 0.38685)n - p)} \\ &= \mathcal{O}(2.6151^n). \end{aligned}$$

To conclude, if  $|C_i| \geq 0.38685n$ , we compute the treewidth of  $G$  in polynomial space within  $\mathcal{O}(2.6151^n)$  steps.  $\blacksquare$

**Acknowledgement.** We are grateful to Saket Saurabh for many useful comments, and to the anonymous referee pointing out that one of the bounds matched the golden ratio.

## References

- [1] S. ARNBORG, D. G. CORNEIL and A. PROSKUROWSKI: Complexity of finding embeddings in a  $k$ -tree, *SIAM J. Algebraic Discrete Methods* **8** (1987), 277–284.
- [2] C. BEERI, R. FAGIN, D. MAIER and M. YANNAKAKIS: On the desirability of acyclic database schemes, *J. ACM* **30** (1983), 479–513.
- [3] A. BERRY, J. P. BORDAT and O. COGIS: Generating all the minimal separators of a graph., *Int. J. Found. Comput. Sci.* **11** (2000), 397–403.
- [4] A. BJÖRKLUND, T. HUSFELDT, P. KASKI and M. KOIVISTO: The travelling salesman problem in bounded degree graphs, in: *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP 2008)*, vol. 5125 of Lecture Notes in Comput. Sci., Springer, 2008, 198–209.
- [5] A. BJÖRKLUND, T. HUSFELDT, P. KASKI and M. KOIVISTO: Trimmed Moebius inversion and graphs of bounded degree, *Theory Comput. Syst.* **47** (2010), 637–654.
- [6] H. L. BODLAENDER: A linear-time algorithm for finding tree-decompositions of small treewidth, *SIAM J. Comput.* **25** (1996), 1305–1317.

- [7] H. L. BODLAENDER: A partial  $k$ -arboretum of graphs with bounded treewidth, *Theor. Comp. Sci.* **209** (1998), 1–45.
- [8] H. L. BODLAENDER, F. V. FOMIN, A. M. C. A. KOSTER, D. KRATSCH and D. M. THILIKOS: On exact algorithms for treewidth, in: *Proceedings of the 14th Annual European Symposium on Algorithms* (ESA 2006), vol. 4168 of Lecture Notes in Comput. Sci., Springer, 2006, 672–683.
- [9] H. L. BODLAENDER and A. M. C. A. KOSTER: Combinatorial Optimization on Graphs of Bounded Treewidth, *The Computer Journal* **51** (3), 2008, 255–269.
- [10] B. BOLLOBÁS: On generalized graphs, *Acta Math. Acad. Sci. Hungar.* (1965), 447–452.
- [11] V. BOUCHITTÉ and I. TODINCA: Treewidth and minimum fill-in: Grouping the minimal separators, *SIAM J. Comput.* **31** (2001), 212–232.
- [12] V. BOUCHITTÉ and I. TODINCA: Listing all potential maximal cliques of a graph., *Theor. Comput. Sci.* **276** (2002), 17–32.
- [13] P. BUNEMAN: A characterization of rigid circuit graphs, *Discrete Math.* **9** (1974), 205–212.
- [14] M. DAVIS, G. LOGEMANN and D. LOVELAND: A machine program for theorem-proving, *Comm. ACM* **5** (1962), 394–397.
- [15] M. DAVIS and H. PUTNAM: A computing procedure for quantification theory, *J. ACM* **7** (1960), 201–215.
- [16] Y. DOURISBOURE and C. GAVOILLE: Tree-decompositions with bags of small diameter *Discrete Mathematics*, **307** (2007), 2008–2029.
- [17] U. FEIGE, M. HAJIAGHAYI and J. R. LEE: Improved approximation algorithms for minimum weight vertex separators, *SIAM J. Comput.* **38** (2008), 629–657.
- [18] F. FOMIN, F. GRANDONI and D. KRATSCH: Some new techniques in design and analysis of exact (exponential) algorithms, *Bulletin of the European Association for Theoretical Computer Science* **87** (2005), 47–77.
- [19] F. V. FOMIN and D. KRATSCH: *Exact Exponential Algorithms*, Springer, 2010.
- [20] F. V. FOMIN, D. KRATSCH and I. TODINCA: Exact algorithms for treewidth and minimum fill-in, in: *Proceedings of the 31st International Colloquium on Automata, Languages and Programming* (ICALP 2004), vol. 3142 of Lecture Notes in Comput. Sci., Springer, Berlin, 2004, 568–580.
- [21] F. V. FOMIN, D. KRATSCH, I. TODINCA and Y. VILLANGER: Exact algorithms for treewidth and minimum fill-in, *SIAM J. Comput.* **38** (2008), 1058–1079.
- [22] F. V. FOMIN and Y. VILLANGER: Treewidth computation and extremal combinatorics, in: *Proceedings of the 34th International Colloquium on Automata, Languages and Programming* (ICALP 2008), vol. 5125 of Lecture Notes in Comput. Sci., Springer, 2008, 210–221.
- [23] M. C. GOLUMBIC: *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, (1980).
- [24] R. L. GRAHAM, D. E. KNUTH and O. PATASHNIK: *Concrete mathematics: A foundation for computer science*, Addison-Wesley Publishing Company, Reading, MA, second ed., 1994.
- [25] M. HELD and R. M. KARP: A dynamic programming approach to sequencing problems, *Journal of SIAM* **10** (1962), 196–210.
- [26] C.-W. HO and R. C. T. LEE: Counting clique trees and computing perfect elimination schemes in parallel, *Inform. Process. Lett.* **31** (1989), 61–68.

- [27] K. IWAMA: Worst-case upper bounds for  $k$ -SAT, *Bulletin of the European Association for Theoretical Computer Science* **82** (2004), 61–71.
- [28] S. JUKNA: *Extremal combinatorics with applications in computer science*, Springer-Verlag, Berlin, 2001.
- [29] T. KLOKS and D. KRATSCHE: Listing all minimal separators of a graph., *SIAM J. Comput.* **27** (1998), 605–613.
- [30] D. LOKSHTANOV: On the complexity of computing treelength, in: *Proceedings of the 32nd International Symposium Mathematical Foundations of Computer Science (MFCS 2007)*, vol. 4708 of Lecture Notes in Comput. Sci., Springer, 2007, 276–287.
- [31] S. PARTER: The use of linear graphs in Gauss elimination, *SIAM Review* **3** (1961), 119–130.
- [32] N. ROBERTSON and P. D. SEYMOUR: Graph minors. II. Algorithmic aspects of tree-width, *Journal of Algorithms* **7** (1986), 309–322.
- [33] D. J. ROSE: A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations, in: *Graph Theory and Computing*, R. C. Read, ed., Academic Press, New York, 1972, 183–217.
- [34] U. SCHÖNING: Algorithmics in exponential time, in: *Proceedings of the 22nd International Symposium on Theoretical Aspects of Computer Science (STACS 2005)*, vol. 3404 of Lecture Notes in Comput. Sci., Springer, 2005, 36–43.
- [35] S. SEIDEN: Theoretical computer science cheat sheet, *SIGACT News* **27** (1996), 52–61.
- [36] R. E. TARJAN and A. E. TROJANOWSKI: Finding a maximum independent set, *SIAM J. Computing* **6** (1977), 537–546.
- [37] Y. VILLANGER: Improved exponential-time algorithms for treewidth and minimum fill-in, in: *Proceedings of the 7th Latin American Theoretical Informatics Symposium (LATIN 2006)*, vol. 3887 of Lecture Notes in Comput. Sci., Springer, 2006, 800–811.
- [38] G. WOEGINGER: Exact algorithms for NP-hard problems: A survey, in: *Combinatorial Optimization – Eureka, You Shrink!*, vol. 2570 of Lecture Notes in Comput. Sci., Springer, 2003, 185–207.

Fedor V. Fomin, Yngve Villanger

*Department of Informatics*

*University of Bergen*

*N-5020 Bergen*

`{fedor.fomin,yngve.villanger}@ii.uib.no`