REPORTS IN INFORMATICS

ISSN 0333-3590

Computing branchwidth via efficient triangulations and blocks

Fedor V. Fomin, Frederic Mazoit, and Ioan Todinca

REPORT NO 297 May 2005

Department of Informatics

UNIVERSITY OF BERGEN

Bergen, Norway

This report has URL
http://www.ii.uib.no/publikasjoner/texrap/ps/2005-297.ps

Reports in Informatics from Department of Informatics, University of Bergen, Norway, is available at http://www.ii.uib.no/publikasjoner/texrap/.

Requests for paper copies of this report can be sent to:

Department of Informatics, University of Bergen, Høyteknologisenteret,
P.O. Box 7800, N-5020 Bergen, Norway

Computing branchwidth via efficient triangulations and blocks

Fedor Fomin* Frédéric Mazoit† Ioan Todinca‡

Abstract

Minimal triangulations and potential maximal cliques are the main ingredients for a number of polynomial time algorithms on different graph classes computing the treewidth of a graph. Potential maximal cliques are also the main engine of the fastest so far $\mathcal{O}(1.9601^n)$-time exact treewidth algorithm. Based on the recent results of Mazoit, we define the structures that can be regarded as minimal triangulations and potential maximal cliques for branchwidth: efficient triangulations and blocks. We show how blocks can be used to construct an algorithm computing the branchwidth of a graph on n vertices in time $(2+\sqrt{3})^n \cdot n^{O(1)}$.

1 Introduction

Treewidth is one of the most basic parameters in graph algorithms and it plays an important role in structural graph theory. Treewidth serves as the main tools in Robertson and Seymour's Graph Minors project [18]. 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. See [3] for a comprehensive survey.

The branchwidth is strongly related to treewidth. It is known that for any graph G, $\mathrm{bw}(G) \leq \mathrm{tw}(G) + 1 \leq 1.5 \cdot \mathrm{bw}(G)$. Both bounds are tight and achievable on trees and complete graphs. Branchwidth was introduced by Robertson & Seymour and it appeared to be even more appropriate tool than treewidth for Graph Minor Theory. Since both parameters are so close, one can expect that the algorithmic behavior of the problems is also quite similar. However, this is not true. For example, on planar graphs branchwidth is solvable in polynomial time [21] while computing the treewidth of a planar graph in polynomial time is a long standing open problem. Even more striking example was observed by Kloks et al. in [14]: it appeared that computing branchwidth is NP hard even on split graphs. Note that the treewidth of a split graph can be found in linear time.

The last decade has led to much research in fast exponential-time algorithms. Examples of recently developed exponential algorithms are algorithms for Maximum Independent Set [13, 19], (Maximum) Satisfiability [7, 12, 17, 20, 23], Coloring [2, 5, 8], and many others (see the recent survey written by Woeginger [24] for an overview). There are several relatively simple algorithms based on dynamic programming computing the treewidth of a graph on n vertices in time $2^n \cdot n^{O(1)}$ which with more careful analyze can be speed-up to $\mathcal{O}(1.9601^n)$ [9]. No such algorithm is known for branchwidth. The only nontrivial algorithm for branchwidth we were aware can be obtained by using deep min-max theorems of Robertson & Seymour [18] relating branchwidth and tangles. Then by playing with tangle

*Department of Informatics, University of Bergen, N-5020 Bergen, Norway, fomin@ii.uib.no.
F. Fomin is supported by Norges forskningsråd projects 162731/V00 and 160778/V30.
†LIF, Université de provence 13453 Marseille Cedex 13 France, Frederic.Mazoit@lif.univ-mrs.fr

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

axioms one can reduce the search space and perform dynamic programming to construct optimal tangles in time $4^n \cdot n^{O(1)}$. (We leave the details in this extended abstract.)

Thus treewidth seems to be more simple problem for design of exponential time algorithms than branchwidth. The explanation to that can be that all known exact algorithms for treewidth exploit the relations between treewidth, minimal triangulations, minimal separators and potential maximal cliques. Mazoit in [15] observed that the branchwidth also can be seen as a triangulation problem. However, while for treewidth one can work only with minimal triangulations the situation with branchwidth is more complicated. Lucky enough we still can use some specific triangulations, which we call efficient triangulations. In this paper we adopt the techniques of Mazoit to discover the analogue of potential maximal cliques for branchwidth, we call these structures by blocks. Potential maximal cliques are extremely useful tools in work with treewidth [4, 9]. We believe that blocks can also be useful to work with branchwidth. To exemplify that we show how blocks can be used to compute branchwidth in time $(2+\sqrt{3})^n \cdot n^{O(1)}$. Note that this is the fastest known exact algorithm for this problem.

2 Basic definitions

We denote by $G = (V, E)$ a finite undirected and simple graph with $|V| = n$ vertices and $|E| = m$ edges. Throughout this paper we use a modified big-Oh notation that suppresses all polynomially bounded factors. For functions f and g we write $f(n) = \mathcal{O}^*\big(g(n)\big)$ if $f(n) = g(n) \cdot n^{\mathcal{O}(1)}$.

For any non-empty subset $W \subseteq V$, the subgraph of G induced by W is denoted by $G[W]$. If S is a set of vertices, we denote by $G - S$ the graph $G[V \setminus S]$. The *neighborhood* of a vertex v is $N(v) = \big\{u \in V : \{u, v\} \in E\big\}$ 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. Let $\omega(G)$ denote the maximum clique size of G.

A graph G is *chordal* if every cycle of G with at least four vertices has a chord, that is an edge between two non-consecutive vertices of the cycle. Consider an arbitrary graph $G = (V, E)$, and a supergraph $H = (V, F)$ of G (i.e. $E \subseteq F$). We say that H is a *triangulation* of G if H is chordal. Moreover, if no strict sub-graph of H is a triangulation of G, then H is called a *minimal triangulation*.

The notion of branchwidth is due to Robertson and Seymour [18]. A *branch decomposition* of a graph $G = (V, E)$ is a pair (T, τ) in which $T = (V_T, E_T)$ is a ternary tree (i.e. each node is of degree one or three) and τ is a function mapping each edge of G on a leaf of T. The vertices of T will be called *nodes* and its edges will be called *branches*. For any branch $e \in E_T$, let $T_1(e)$ and $T_2(e)$ be the subtrees obtained from T by removing e. Let $\mathrm{lab}(e)$ be the set of vertices of G both incident to edges mapped on $T_1(e)$ and $T_2(e)$. The maximum of $\big\{|\,\mathrm{lab}(e)|,\, e \in E_T\big\}$, is called the *width* of the branch decomposition. The *branchwidth* of a graph G $\big(\mathrm{bw}(G)\big)$ is the minimum width over all branch decompositions of G. Note that the definitions of branch decomposition and branch-width also apply to hypergraphs. As pointed by Robertson and Seymour, the definition of branch decomposition can be relaxed. A *relaxed branch decomposition* of $G = (V, E)$ is a couple (T, τ) where T is an arbitrary tree and τ is an application mapping each edge of G to at least one leaf of T. The labels of the branches and the width of the decomposition are defined as before. From any relaxed branch decomposition we can construct a branch decomposition without increasing the width.

The branchwidth is strongly related to a well-known graph parameter introduced by Robertson and Seymour, namely the *treewidth*. One of the equivalent definitions for treewidth is $\mathrm{tw}(G) = \min\big\{\omega(H) - 1 \mid H \text{ is a triangulation of } G\big\}$. Robertson and Seymour show that the two parameters differ by at most a factor of 1.5. More precisely, for any graph G we have $\mathrm{bw}(G) \le \mathrm{tw}(G) + 1 \le 1.5\,\mathrm{bw}(G)$. In particular, if G is a complete graph, its treewidth is $n - 1$, while its branchwidth is $\lceil 2n/3 \rceil$ (see [18]). Clearly, when computing the treewidth of a graph we can restrict to minimal triangulations. This observation and the study of minimal triangulations of graphs led to several results about treewidth computation, including an exact algorithm in $\mathcal{O}^*(1.961^n)$ time.

The branch decompositions of a graph can also be associated to triangulations. Indeed, given a branch decomposition (T, τ) of $G = (V, E)$, we can associate to each $x \in V$ the subtree of T covering all the leaves of T containing edges incident to x. It is well-known that the intersection graph of the sub-trees of a tree is chordal [10]. Thus the intersection graph of the trees T_x is a triangulation $H(T, \tau)$ of G. Note that for each branch $e \in E_T$, $\mathrm{lab}(e)$ is the set of vertices x such that e belongs to T_x. In particular, $\mathrm{lab}(e)$ induces a clique in $H(T, \tau)$, not necessarily maximal. (We shall point out later that, for each maximal clique Ω of $H(T, \tau)$, there exists a node u of T such that $u \in T_x$ for all $x \in \Omega$.)

The first big difference with treewidth is that there exist examples of graphs for which any optimal branch decomposition leads to non-minimal triangulations [15]. Therefore the many existing tools about minimal triangulations are not sufficient in our case. The second important difference is that the branchwidth problem remains NP-hard even for a restricted class of chordal graphs, the *split* graphs [14]. Nevertheless, our technique for computing the branchwidth relies on a structural result stating that, for any graph G, there is an optimal branch decomposition (T, τ) such that $H(T, \tau)$ is an *efficient* triangulation of G. The efficient decomposition, defined in the next section, behave somehow similarily to minimal decompositions. In order to obtain our exact algorithm for branchwidth, we will combine this observation with an exponential algorithm computing the branchwidth of hyper-cliques.

3 Branchwidth and efficient triangulations

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 in the graph $G - S$ a and b in are in different connected components. 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. We denote by $\mathcal{C}(S)$ the set of connected components of $G - S$ and by Δ_G the set of all minimal separators of G.

Definition 1. A triangulation H of G is *efficient* if

1. each minimal separator of H is also a minimal separator of G;
2. for each minimal separator S of H, the connected components of $H - S$ are exactly the connected components of $G - S$.

In particular, all the minimal triangulations of G are efficient [16].

Theorem 2 ([15]). *There is an optimal branch decomposition (T, τ) of G such that the chordal graph $H(T, \tau)$ is an efficient triangulation of G. Moreover, each minimal separator of H is the label of some branch of T.*

Definition 3. A set of vertices $B \subseteq V$ of G is called a *block* if, for each connected component C_i of $G - B$,

- its neighborhood $S_i = N(C_i)$ is a minimal separator;
- $B \setminus S_i$ is non empty and contained in a connected component of $G - S_i$.

We say that the minimal separators S_i *border* the block B and we denote by $\mathcal{S}(B)$ the set of these separators.

Let $\mathcal{B}_G$ denote the set of blocks of G. Note that V is a block with $\mathcal{S}(V) = \emptyset$.

We prove that if H is an efficient triangulation of G, then any maximal clique K of H is a block of G.

Lemma 4 ([4]). *Let H be a chordal graph and Ω be a maximal clique of H. Then Ω is a block of H.*

Lemma 5. *Let H be an efficient triangulation of G and Ω be any maximal clique of H. Then Ω is a block of G. Conversely, for any block B of G, there is an efficient triangulation $H(B)$ of G such that B induces a maximal clique in H.*

Proof. If H is an efficient triangulation of G, by Lemma 4 every maximal clique Ω is a block of H. By definition of efficient triangulations, a block of H is also a block of G.

Conversely, if B is a block of G, let $C_1, \dots, C_p$ be the connected components of $G - B$ and let $S_i = N(C_i)$, for all $1 \leq i \leq p$. Let $H(B)$ be the graph obtain from G by completing B and each set $S_i \cup C_i$ into a clique. The minimal separators of $H(B)$ are exactly $S_1, \dots, S_p$. Moreover, for each S_i, the connected components of $H - S_i$ are exactly the components of $G - S_i$. □

Note that the treewidth of a graph can be expressed by the following equation:

$$\mathrm{tw}(G) = \min_{H \text{ triangulation of } G} \max\{|\Omega| - 1 \mid \Omega \text{ maximal clique of } H\}. \quad (1)$$

The minimum can be taken over all minimal triangulations H of G. A similar formula can be obtained for branchwidth.

Definition 6 (block-branchwidth). Let B be a block of G and $K(B)$ be the complete graph with vertex set B. A branch decomposition (T_B, τ_B) of $K(B)$ *respects* the block B if, for each minimal separator $S \in \mathcal{S}(B)$, there is a branch e of the decomposition such that $S \subseteq \mathrm{lab}(e)$. The *block branchwidth* $\mathrm{bbw}(B)$ of B is the minimum width over all the branch decompositions of $K(B)$ respecting B.

Equivalently, $\mathrm{bbw}(B)$ is the branchwidth of the hypergraph obtained from the complete graph with vertex set B by adding a hyperedge S for each minimal separator S bordering B. The block-branchwidth allows us to express the branchwidth of G by a formula similar to Equation 1 (see Propositions 4.18 and 6.7 in [15]). The proof of the theorem is given in the Appendix.

Theorem 7 ([15]).

$$\mathrm{bw}(G) = \min_{H \text{ efficient triangulation of } G} \max\{\mathrm{bbw}(\Omega) \mid \Omega \text{ maximal clique of } H\}. \quad (2)$$

A potential maximal clique of a graph G is a set of vertices Ω such that there is a minimal triangulation H of G in which Ω introduces a maximal clique [4]. Using the Equation 1, Bouchitté and Todinca show that, given a graph and all its potential maximal cliques, the treewidth of the graph can be computed in polynomial time. The result is refined in [9], where the authors show the following:

Theorem 8. *There is an algorithm that, given a graph G and the set Π_G of its potential maximal cliques, computes the treewidth of G in $\mathcal{O}(nm|\Pi_G|)$ time.*

According to Lemma 5, a vertex subset Ω of G can be a maximal clique of an efficient triangulation H of G if and only if Ω is a block of G. Hence, in our case the blocks play the same role as the potential maximal cliques in Theorem 8.

Using Equation 2 instead of Equation 1 and blocks instead of potential maximal cliques, the algorithm cited in Theorem can be directly transformed into an algorithm taking G, the set $\mathcal{B}_G$ of all its blocks and the block-branchwidth of each block B, and computing the branchwidth of G in $\mathcal{O}(nm|\mathcal{B}_G|)$ time. In the rest of this section we give, without proofs, the new algorithm and the main tools for obtaining it.

Given a minimal separator S of G and a connected component C of $G - S$, let $R(S, C)$ denote the hypergraph obtained from $G[S \cup C]$ by adding the hyperedge S.

Lemma 9 (Similar to Corollary 4.5 in [4]). *For any graph G,*

$$\mathrm{bw}(G) = \min(\lceil 2n/3 \rceil, \min_{S \in \Delta_G} \max_{C \in \mathcal{C}(S)} \mathrm{bw}(R(S,C)))$$

Moreover, the minimum can be taken over the inclusion-minimal separators of G.

The case when $\mathrm{bw}(G) = \lceil 2n/3 \rceil$ corresponds to the fact that, for an optimal decomposition (T,τ) of G, the efficient triangulation $H(T,\tau)$ is the complete graph.

Lemma 10 (Similar to Corollary 4.8 in [4]). *Let S be a minimal separator of G and C be a component of $G - S$ such that $S = N(C)$. Then*

$$\mathrm{bw}(R(S,C)) = \min_{\textit{blocks } \Omega \textit{ s.t. } S \subset \Omega \subseteq S \cup C} \max(\mathrm{bbw}(\Omega), \mathrm{bw}(R(S_i, C_i)))$$

where C_i are the components of $G - \Omega$ contained in C and $S_i = N(C_i)$.

The algorithm for computing the branchwidth of G is a straightforward translation of Lemmas 9 and 10, and very similar to the one of [9].

```
Input: G, all its blocks and all its minimal separators
Output: bw(G)
begin
    compute all the pairs {S, C} where S is a minimal separator and C a component
        of G − S with S = N(C); sort them by the size of S ∪ C
    for each {S, C} taken in increasing order
        bw(R(S, C)) := bbw(S ∪ C)
        for each block Ω with S ⊂ Ω ⊆ S ∪ C
            compute the components C_i of G − Ω contained in C and let S_i = N(C_i)
            bw(R(S, C)) := min(bw(R(S, C)),
                                max_i(bbw(Ω), bw(R(S_i, C_i))))
        end_for
    end_for
    let Δ*_G be the set of inclusion-minimal separators of G
    bw(G) := min(⌈2n/3⌉, min_{S∈Δ*_G} max_{C∈C(S)} bw(R(S, C)))
end
```

Theorem 11. *Given a graph G and the list $\mathcal{B}_G$ of all its blocks together with their block-branchwidth, the branchwidth of G can be computed in $\mathcal{O}(nm|\mathcal{B}_G|)$ time.*

Proof. The proof is very similar to the proof for treewidth and potential maximal cliques in [9] and we omit it here. □

4 Computing the block-branchwidth

The main result of this section is that the block-branchwidth of a block B of G can be computed in $\mathcal{O}^*(\sqrt{3}^n)$ time. Computing the block-branchwidth is NP-hard, as it can be deduced directly from [14].

Let $n(B)$ denote the number of vertices of the block B of G and let $s(B)$ be the number of minimal separators bordering B. Note that $s(B)$ is at most the number of components of $G - B$, in particular $n(B) + s(B) \leq n$.

Lemma 12. $\mathrm{bbw}(B) \leq p$ *if and only if there is a partition of B into four parts A_1, A_2, A_3, D such that*

1. $|B \setminus A_i| \leq p$*, for all* $i \in \{1,2,3\}$*;*

2. *for each minimal separator $S \in \mathcal{S}(B)$, S is contained in $B \setminus A_i$ for some $i \in \{1, 2, 3\}$.*

Proof. Suppose that $\text{bbw}(B) \le p$ and let (T, τ) be an optimal branch decomposition of B respecting the block. Recall that this branch decomposition corresponds to the complete graph $K(B)$ with vertex set B. For each $x \in B$ let T_x be the minimal sub-tree of T spanning all the leaves of T labeled with an edge incident to x. Let u represent B. Clearly u is a ternary node. Let e_1, e_2, e_3 be the branches of T incident to u. Let $T(i)$ be the sub-tree of T rooted in u, containing the branch e_i, for $i \in \{1, 2, 3\}$. Let $B_i = \{z \in B \mid z \text{ is incident to some edge of } K(B) \text{ mapped on a leaf of } T(i)\}$. Fix $D = B_1 \cap B_2 \cap B_3$, and $A_i = B_j \cap B_k \setminus D$ for all triples (i, j, k) with $i, j, k \in \{1, 2, 3\}$ and distinct. Observe that D, A_1, A_2, A_3 form a partition of B. The three sets are pairwise disjoint by construction. Since for all $x \in B, u \in T_x$, we have that $x \in B_i \cap B_j$ for distinct $i, j \in \{1, 2, 3\}$, so x is in one of the four sets A_1, A_2, A_3 or D. It remains to show that the partition satisfies the conditions of the theorem. Consider a separator $S \in \mathcal{S}(B)$ and a branch e in the decomposition with $S \subseteq \text{lab}(e)$. Suppose w.l.o.g. that $e \in T(i)$. Consequently $\text{lab}(e) \subseteq B_i$, and since $B_i = B \setminus A_i$ we have the second condition of the theorem. For proving the first condition, since A_1, A_2, A_3, D is a partition of B, note that $\text{lab}(e_i) = A_j \cup A_k \cup D = B \setminus A_i$. Therefore $|B \setminus A_i| \le p$, for all $i \in \{1, 2, 3\}$.

Conversely, suppose that such a partition exists and let us construct a branch decomposition of $K(B)$ respecting the block B, of width at most p. Let $B_i = B \setminus A_i$, for each $i \in \{1, 2, 3\}$. For each i, construct an arbitrary branch decomposition (T_i, τ_i) of the complete graph with vertex set B_i. Let T be the tree obtained as follows : for each T_i, add a new node v_i on some branch of T_i, then glue the three trees by adding a new node u, adjacent to v_1, v_2, v_3. The tree T is a ternary tree and each edge of $K(B)$ is mapped on at least one leaf of T, so we obtained a relaxed tree decomposition (T, τ) of $K(B)$. Let e_i be the branch $\{u, v_i\}$. Note that $\text{lab}(e_i) = B_i \cap (B_j \cup B_k)$, where $\{i, j, k\} = \{1, 2, 3\}$. Hence $\text{lab}(e_i) = B_i$. Consequently, the relaxed branch decomposition respects the block B. Clearly for each branch e of T, $\text{lab}(e)$ is contained in some B_i, so $|\text{lab}(e)| \le p$ and the conclusion follows. ☐

Theorem 13. *The block-branchwidth of any block B can be computed in $\mathcal{O}^*(3^{s(B)})$ time.*

Proof. Let B be a block of G. Suppose that $\text{bbw}(B) \le p$. By Lemma 12, there exists a partition of B in A_1, A_2, A_3 and D such that $|B \setminus A_i| \le p$ and every $S \in \mathcal{S}(B)$ is a subset of $B \setminus A_i$. Denote by a_1, a_2, a_3 and d the sizes of A_1, A_2, A_3 and D. We can partition $\mathcal{S}(B)$ in three subsets $\mathcal{S}_i$ such that every $S \in \mathcal{S}_i$ is included in $B \setminus A_i$. Let S_i be the union of all the minimal separators of $\mathcal{S}_i$. The numbers a_1, a_2, a_3 and d satisfy the following inequalities:

1. $a_i \ge 0, d \ge 0, a_1 + a_2 + a_3 + d = |B|$;
2. $|S_1 \cap S_2 \cap S_3| \le d$, $|(S_1 \cap S_2) \setminus S_3| \le a_3$, $|(S_2 \cap S_3) \setminus S_1| \le a_1$, $|(S_3 \cap S_1) \setminus S_2| \le a_2$;
3. $a_1 + a_2 + d \le p$, $a_2 + a_3 + d \le p$, $a_3 + a_1 + d \le p$.

The first inequalies express the fact that A_1, A_2, A_3 and D is a partition of B, the second express the fact that S_i is a subset of $B \setminus A_i$ and the last ones express the fact that $\text{bbw}(B) \le p$.

Conversely, suppose there is a partition of $\mathcal{S}(B)$ in $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$ and four integers a_1, a_2, a_3, d satisfying the system above. Then there exist a partition of B into four sets A_1, A_2, A_3, D, of cardinalities a_1, a_2, a_3, d and such that D intersects $S_1 \cup S_2 \cup S_3$ exactly in $S_1 \cap S_2 \cap S_3$, and each A_i intersects $S_1 \cup S_2 \cup S_3$ exactly in $(S_j \cap S_k) \setminus S_i$, where $\{i, j, k\} = \{1, 2, 3\}$. Moreover $|B \setminus A_i| \le p$ by the third series of inequalities, so by Lemma 12 we have $\text{bbw}(B) \le p$.

Hence, there an efficient branch decomposition of $K(B)$ respecting B of branchwidth at most p if and only if there is a partition partition $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ of $\mathcal{S}(B)$ and four numbers

a_1, a_2, a_3 and d satisfying the system. To decide whether $\mathrm{bbw}(B) \leq p$ or not, we only have to try all the partitions of $\mathcal{S}(B)$ in $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$ and check all the n^4 possible values for the a_i's and d. This can be done in $\mathcal{O}^*(3^{|\mathcal{S}(B)|}) = \mathcal{O}^*(3^{s(B)})$ time as claimed. $\square$

Theorem 14. *The block-branchwidth of any block B can be computed in $\mathcal{O}^*(3^{n(B)})$ time.*

Proof. We show that for any number p, the existence of a partition like in Lemma 12 can be tested in $\mathcal{O}^*(3^{n(B)})$.

For this purpose, instead of partitioning B into four parts, we try all the partitions of B into three parts A_1, X, D, where X corresponds to $A_2 \cup A_3$. If $|B \setminus A_1| \leq p$, we check in polynomial time if X can be partitioned into A_2 and A_3 as required. Since there are at most $3^{n(B)}$ three-partitions of B, it only remains to solve this last point.

We say that two vertices $x, y \in X$ are equivalent if there exist $z \in A_1$ and a minimal separator S bordering B such that $x, y, z \in S$. In particular, $x \sim y$ implies that x and y must be both in A_2 or both in A_3. Let $X_1, \ldots, X_q$ be the equivalence classes of X. Then X can be partitioned into A_2 and A_3 as required if and only if $\{|X_1|, \ldots, |X_q|\}$ can be partitioned into two parts of sum at most $p - |A_1| - |D|$ vertices. Consider now the EXACT SUBSET-SUM problem, whose instance is a set of positive integers $I = \{i_1, \ldots, i_q\}$ and a number t, and the problem consists in finding a subset of I whose sum is exactly t. Though NP-hard in general, it becomes polynomial when t and the numbers i_j are polynomially bounded in n (see e.g. the chapter on approximation algorithms, the subset-sum problem in the book of Cormen, Leiserson, Rivest [6]). By taking $I = \{|X_1|, \ldots, |X_q|\}$ and trying all possible values of t between 1 and n^2, we can check in polynomial time if X can be partitioned as required. $\square$

Since at least one of $s(B)$ or $n(B)$ is smaller or equal to $n/2$, we deduce:

Theorem 15. *For any block B of G, the block-branchwidth of B can be computed in $\mathcal{O}^*(\sqrt{3}^n)$ time.*

Theorems 11 and 15 imply our main result.

Theorem 16. *The branchwidth of graphs can be computed in $\mathcal{O}^*((2+\sqrt{3})^n)$ time and $\mathcal{O}^*(2^n)$ space.*

Proof. The algorithm enumerates every subset B of V and checks if B is a block. Clearly, we can verify if B is a block in polynomial time. If so, we compute the block branchwidth of B using Theorem 15. The number of blocks is at most 2^n and for each block we need $\mathcal{O}^*(\sqrt{3}^n)$ for computing its block branchwidth. Hence the running time of this phase is $\mathcal{O}^*((2+\sqrt{3})^n)$, and the space is $\mathcal{O}^*(2^n)$.

Eventually, we use Theorem 11 for computing the branchwidth of G. The second phase takes $\mathcal{O}^*(2^n)$ time and space. $\square$

5 Open problems

Our algorithm is based on the enumeration of the blocks of a graph (in $\mathcal{O}^*(2^n)$ time) and on the computation of the block-branchwidth of a block (in $\mathcal{O}^*(\sqrt{3}^n)$ time). It is natural to ask whether one of these steps can be improved.

Computing the block-branchwidth is the same problem as computing the branchwidth of a complete hypergraph with n' vertices and s' hyper-edges of cardinality at least three. Can we obtain an algorithm faster than our $O(\max(3^{n'}, 3^{s'}))$-time algorithm?

Note that there exist graphs with n vertices having $2^n/n^{O(1)}$ blocks. Indeed, consider the disjoint union of a clique K and an independent set I, both having $n/2$ vertices, and add a perfect matching between K and I. We obtain a graph G_n such that for any $I' \subseteq I$, $G_n - I'$ is a block. Thus G_n has at least $\binom{n}{n/2} \geq 2^n/n$ blocks. The interesting question here is if we can define a new class of triangulations, smaller than the efficient triangulations but also containing $H(T, \tau)$ for some optimal branch decompositions of the graph.

References

[1] R. Beigel and D. Eppstein. 3-coloring in time $O(1.3446^n)$: a no-MIS algorithm. Proceedings of the *36th IEEE Symposium on Foundations of Computer Science (FOCS 1995)*, pp. 444–452.

[2] R. Beigel and D. Eppstein. 3-coloring in time $O(1.3289^n)$. *Journal of Algorithms,* 54:444–453, 2005.

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

[4] V. Bouchitté and I. Todinca. Treewidth and minimum fill-in: grouping the minimal separators. *SIAM J. on Computing*, 31(1):212 – 232, 2001.

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

[6] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to algorithms*. The MIT press, 1990.

[7] 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. *Theoretical Computer Science*, 289(1):69–83, 2002.

[8] D. Eppstein. Improved algorithms for 3-coloring, 3-edge-coloring, and constraint satisfaction. Proceedings of the *12th ACM-SIAM Symposium on Discrete Algorithms (SODA 2001)*, pp. 329–337.

[9] F. Fomin, D. Kratsch, and I. Todinca. Exact (exponential) algorithms for treewidth and minimum fill-in. In *Proceedings 31st International Colloquium on Automatas, Languages and Programming (ICALP'04)*, volume 3142 of *Lecture Notes in Computer Science*, pages 568–580. Springer, 2004.

[10] F. Gavril. The intersection graphs of a path in a tree are exactly the chordal graphs. *Journal of Combinatorial Theory*, 16:47–56, 1974.

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

[12] K. Iwama and S. Tamaki. Improved upper bounds for 3-SAT. Proceedings of the *15th ACM-SIAM Symposium on Discrete Algorithms (SODA 2004)*, p.328.

[13] T. Jian. An O$(2^{0.304n})$ algorithm for solving maximum independent set problem. *IEEE Transactions on Computers*, 35(9):847–851, 1986.

[14] T. Kloks, J. Kratochvíl, and H. Müller. New branchwidth territories. In *Proceedings 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS '99)*, volume 1563 of *Lecture Notes in Computer Science*, pages 173–183. Springer, 1999.

[15] F. Mazoit. *Décompositions algorithmiques des graphes*. PhD thesis, École normale supérieure de Lyon, 2004. In French.

[16] A. Parra and P. Scheffler. Characterizations and algorithmic applications of chordal graph embeddings. *Discrete Appl. Math.*, 79(1-3):171–188, 1997.

[17] R. Paturi, P. Pudlak, M. E. Saks, and F. Zane. An improved exponential-time algorithm for k-SAT. Proceedings of the *39th IEEE Symposium on Foundations of Computer Science (FOCS 1998)*, pp. 628–637.

[18] N. Robertson and P. Seymour. Graph minors X. Obstructions to tree decompositions. *Journal of Combinatorial Theory Series B*, 52:153–190, 1991.

[19] J. M. Robson. Algorithms for maximum independent sets. *Journal of Algorithms*, 7(3):425–440, 1986.

[20] U. Schoning. A Probabilistic Algorithm for k-SAT and Constraint Satisfaction Problems. Proceedings of the *40th IEEE Symposium on Foundations of Computer Science (FOCS 1999)*, pp. 410-414.

[21] P. D. Seymour and R. Thomas, Call routing and the ratcatcher, *Combinatorica*, 14:217–241, 1994.

[22] R. Tarjan and A. Trojanowski. Finding a maximum independent set. *SIAM Journal on Computing*, 6(3):537–546, 1977.

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

[24] G. J. Woeginger. Exact algorithms for NP-hard problems: A survey. *Combinatorial Optimization – Eureka, You Shrink*, Springer LNCS vol. 2570, 2003, pp. 185–207.

A Appendix

Theorem 7 ([15]).

$$\mathrm{bw}(G) = \min_{H \textit{ efficient triangulation of } G} \max\{\mathrm{bbw}(\Omega) \mid \Omega \textit{ maximal clique of } H\}. \quad (2)$$

Proof. Let (T, τ) be an optimal branch decomposition of G such that $H = H(T, \tau)$ is an efficient triangulation of G. Such a decomposition exists by Theorem 2. First, let us construct a branch decomposition (T', τ') of H having the same width as (T, τ). For each edge $\{x, y\}$ of $E(H) - E(G)$, the sub-trees T_x and T_y share a branch e. We divide the branch e by a node v, add a leaf w adjacent to v and map the edge $\{x, y\}$ on w. Clearly this will not increase the width of the decomposition. Consider any maximal clique Ω of G. By Lemma 5, Ω is a block of G and by Theorem 2 each minimal separator bordering Ω is contained in the label of some branch e_S of T'. For each S let (T_S, τ_S) be a arbitrary branch decomposition of the clique $K(S)$. We glue this decomposition to T' on the branch e_S. That is, we add a node on e_S and a node on some branch of T_S and make them adjacent. We call this new edge e'_S, in particular its label is exactly S. By this process we obtain a relaxed branch decomposition (T'', τ'') of H of same width as (T', τ'). By removing from T'' all the leaves that do not correspond to edges in the clique Ω, we obtained a relaxed clique decomposition of the complete graph $K(\Omega)$. For each minimal separator S bordering Ω, note that S is contained in the label of the edge e'_S, so the new decomposition respects Ω. Hence $\mathrm{bbw}(\Omega) \leq \mathrm{bw}(G)$ for each maximal clique Ω of H.

Conversely, let H be any efficient triangulation of G, let us show that $\mathrm{bw}(G) \leq \max\{\mathrm{bbw}(\Omega) \mid \Omega \text{ maximal clique of } H\}$. For each maximal clique Ω of G, let (T_Ω, τ_Ω) be an optimal branch decompoition of $K(\Omega)$, respecting the block Ω. We connect these decompositions into a relaxed branch decomposition of H. For this purpose we use a *clique tree* associated to the chordal graph graph H (see e.g. [11]). A clique tree is given by a tree $T = (V_T, E_T)$ and a one-to-one correspondence between the nodes of T and the maximal cliques of H such that, for each Ω, Ω' maximal cliques of H, their intersection is contained in all the cliques associated to nodes on the unique path from u_Ω to $u_{\Omega'}$ of T (u_Ω and $u_{\Omega'}$ denote the nodes associated to Ω and Ω' respectively). Moreover, for each branch $e = \{u_\Omega, u_{\Omega'}\}$ of T, $S = \Omega \cap \Omega'$ is a minimal separator bordering Ω and Ω' [11]. Let e_S (resp. e'_S) be a branch of T_Ω (resp. $T_{\Omega'}$) whose label contains S. We connect T_Ω and $T_{\Omega'}$ by adding a new branch between the middle of e_S and e'_S, for all branches $\{u_\Omega, u_{\Omega'}\}$ of T. Hence we obtain a relaxed branch decomposition of H. By the properties of the clique tree, the label of each newly created edge connecting T_Ω and $T_{\Omega'}$ is exactly $S = \Omega \cap \Omega'$. Consequently, the labels of the branches contained in some T_Ω do not change. Hence $\mathrm{bw}(H) \leq \max\{\mathrm{bbw}(\Omega) \mid \Omega \text{ maximal clique of } H\}$. G being a sub-graph of H, we have $\mathrm{bw}(G) \leq \mathrm{bw}(H)$ and the conclusion follows. $\square$