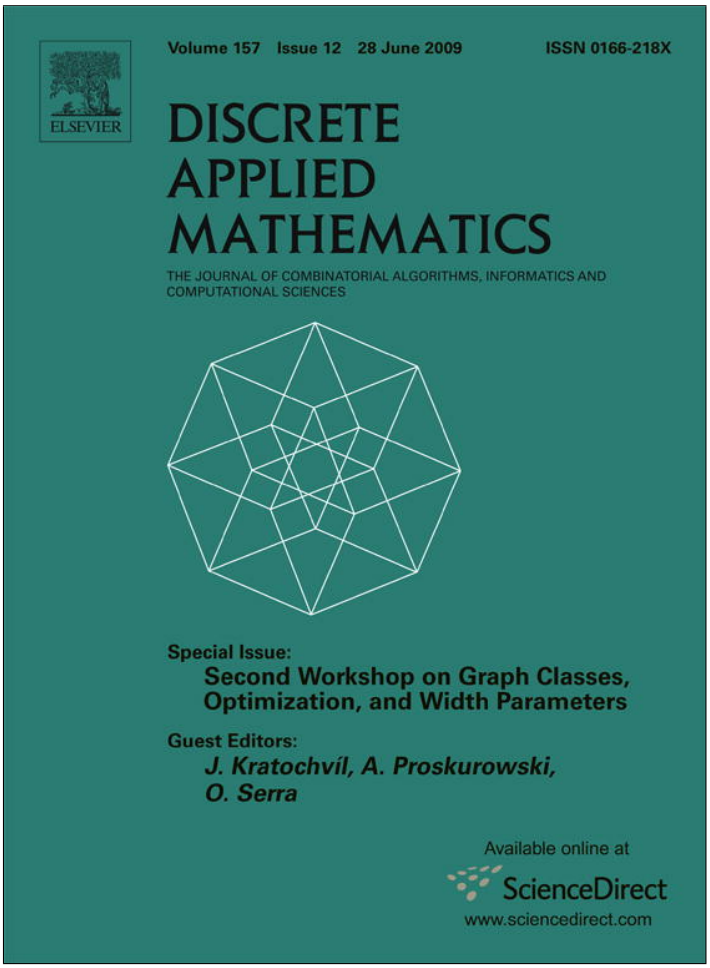
Provided for non-commercial research and education use.
Not for reproduction, distribution or commercial use.



This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

http://www.elsevier.com/copyright

Author's personal copy

Discrete Applied Mathematics 157 (2009) 2726–2736

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Computing branchwidth via efficient triangulations and blocks

Fedor V. Fomin [a], Frédéric Mazoit [b,*,1], Ioan Todinca [c]

[a] Department of Informatics, University of Bergen, PO Box 7800, 5020 Bergen, Norway
[b] LaBRI Université Bordeaux F-33405 Talence cedex, France
[c] LIFO Université d'Orléans 45067 Orléans cedex 2, France

ARTICLE INFO

Article history:
Received 13 February 2006
Received in revised form 29 March 2007
Accepted 6 August 2008
Available online 27 September 2008

Keywords:
Graph
Branchwidth
Exponential time algorithm

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, exact (exponential) 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 $(2\sqrt{3})^n \cdot n^{O(1)}$.

© 2009 Published by Elsevier B.V.

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 tool in Robertson and Seymour's Graph Minors project [27]. 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.

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 an even more appropriate tool than treewidth for Graph Minor Theory. Branchwidth was also used in algorithms solving TSP [7], SAT [1], and parameterized algorithms [11,10,14,15].

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 [30] while computing the treewidth of a planar graph in polynomial time is a long standing open problem. An even more striking example was observed by Kloks et al. in [21]. It appeared that computing branchwidth is NP hard, even on split graphs, while the treewidth of a split graph can be found in linear time.

The algorithmic behavior of branchwidth on different graph classes has been much less investigated than treewidth. The main reason for this is that the powerful machinery developed for study treewidth and including minimal triangulations, minimal separators, potential maximal cliques, cannot be applied to branchwidth. As we already mentioned, the branchwidth of a planar graph can be found in polynomial time [30] (see also [18]) but is NP complete on general graphs. Later Kloks et al. [21] showed that branchwidth is NP complete when restricted to split and bipartite graphs, but is computable in polynomial time on interval graphs. (See also the recent work of Paul and Telle [26].) Recently Mazoit [23] proved that the branchwidth of a circular arc graph can be solved in polynomial time.

In his Ph.D. thesis [22], Mazoit investigated how the machinery that worked fine for treewidth (minimal triangulations and potential maximal cliques) can be modified to be used for branchwidth. Based on the approach from [22], we developed a

* Corresponding author. Fax: +33 5 40 00 66 69.
E-mail addresses: fomin@ii.uib.no (F.V. Fomin), Frederic.Mazoit@labri.fr (F. Mazoit), Ioan.Todinca@univ-orleans.fr (I. Todinca).

[1] F. Mazoit received additional support by the ANR projet GRAAL.

0166-218X/$ – see front matter © 2009 Published by Elsevier B.V.
doi:10.1016/j.dam.2008.08.009

Author's personal copy

number of structural results to design an exact (exponential) algorithm for branchwidth. Let us remark, that independently, Paul and Telle [26] have initiated research targeting similar goals, and obtained a number of structural results that are similar to ours. In particular, the notion of k-troika [26] is similar to the result on block-branchwidth obtained in Lemma 20. Paul and Telle use k-troika to obtain a faster (and simpler) algorithm for interval graphs, and show how to generalize such an algorithm to chordal graphs with clique trees having a polynomial number of sub-trees.

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 [20,28,12], (Maximum) Satisfiability [9,19,25,29,32], Coloring [2,6], and many others (see the recent survey written by Woeginger [33] 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 analysis can be sped-up to $\mathcal{O}(1.89^n)$ [13,31]. No such algorithm is known for branchwidth.

Thus treewidth seems to be a more simple problem for the design of exponential time algorithms than branchwidth. The explanation for that is again, that all known exact algorithms for treewidth exploit the relations between treewidth, minimal triangulations, minimal separators and potential maximal cliques. 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. Luckily enough, we still can use some specific triangulations, which we call efficient triangulations. The efficient triangulations were first used, under a different name, in [4]. In this paper we define the analogue of potential maximal cliques for branchwidth, we call these structures *blocks*. We believe that blocks can be useful to work with branchwidth, in the same way as potential maximal cliques for treewidth [5,13]. To exemplify that, we show how blocks can be used to compute the branchwidth of a graph on n vertices in time $(2\sqrt{3})^n \cdot n^{O(1)}$. Note that this is the fastest known exact algorithm for this problem.

The paper is organized as follows. After giving the basic definitions, we introduce in Section 3 the notions of efficient triangulations and block-branchwidth. They allow us to characterize the branchwidth by a formula very similar to one of the classical definitions for treewidth. Using this result, in Section 4 we adapt an algorithm initially designed for treewidth, for the computing of branchwidth. Section 5 is devoted to the computation of block-branchwidth. Eventually, we discuss some open questions.

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 $f(n) = \mathcal{O}^*(g(n))$ 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 $N(v) = \{u \in V : \{u, v\} \in E\}$ 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. 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 [27]. 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*. Let $T_1(e)$ and $T_2(e)$ be the sub-trees of T obtained from T by removing $e \in E_T$. We define the *middle set* of e, $\mathrm{mid}(e)$, as the set of vertices of G both incident to edges mapped on leaves of $T_1(e)$ and $T_2(e)$. The maximum of $\{|\mathrm{mid}(e)|, e \in E_T\}$, is called the *width* of the branch decomposition and is denoted by $\mathrm{width}(T, \tau)$. The *branchwidth* of a graph G ($\mathrm{bw}(G)$) 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 [27], the definition of branch decomposition can be relaxed. A *relaxed branch decomposition* of $G = (V, E)$ is a pair (T, τ) where T is an arbitrary tree of maximum vertex degree at most three and τ maps each edge of G to at least one leaf of T. The middle sets of the branches and the width of the decomposition are defined as before. From any relaxed branch decomposition, one can construct a branch decomposition of the same graph 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 the treewidth of a graph G, $\mathrm{tw}(G)$, is

$$\mathrm{tw}(G) = \min\{\omega(H) - 1 \mid H \text{ is a triangulation of } G\}.$$

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 $\mathrm{bw}(G) \leq \mathrm{tw}(G) + 1 \leq 1.5\mathrm{bw}(G)$. In particular, if G is a complete graph, its treewidth is $n - 1$, while its branchwidth is $\lceil 2n/3 \rceil$ (see [27]). 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 $\mathcal{O}^*(1.89^n)$ time.

Author's personal copy

The branch decompositions of a graph can also be associated with triangulations. Indeed, given a branch decomposition (T, τ) of $G = (V, E)$, we associate with each $x \in V$ the sub-tree T_x 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 [16]. Thus the intersection graph of the trees T_x is a triangulation of G. We denote such a triangulation by $H(T, \tau)$. Note that for each branch $e \in E_T$, $\mathrm{mid}(e)$ is the set of vertices x, such that e belongs to T_x. In particular, $\mathrm{mid}(e)$ 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 $x \in \Omega$.)

Lemma 1. *Let (T, τ) be a branch decomposition of G. There is a branch decomposition (T', τ') of $H(T, \tau)$ such that:*

(1) *T is a sub-tree of T'.*
(2) *For each branch of T, its middle sets in (T, τ) and (T', τ') are equal.*
(3) $\mathrm{width}(T', \tau') = \mathrm{width}(T, \tau)$.

In particular, if (T, τ) is an optimal branch-decomposition of G, we have $\mathrm{bw}(H(T, \tau)) = \mathrm{bw}(G)$.

Proof. If the width of (T, τ) is at most one, then every edge of G has at least one endpoint of degree one. In this case, it is easy to show that $G = H(T, \tau)$, and thus we put $(T', \tau') = (T, \tau)$.

Suppose that the width of (T, τ) is at least two. For each edge $\{x, y\}$ of $E(H(T, \tau)) \setminus E(G)$, the sub-trees T_x and T_y have a branch e in common. We divide the branch e by a node v (i.e. we put on e a vertex v of degree two), add a leaf w adjacent to v and map the edge $\{x, y\}$ on w. Since the width of (T, τ) is at least two, this does not increase the width.

If (T, τ) is optimal, then $\mathrm{bw}(H(T, \tau)) \leq \mathrm{bw}(G)$. Conversely, since G is a sub-graph of $H(T, \tau)$, $\mathrm{bw}(G) \leq \mathrm{bw}(H(T, \tau))$ and Lemma follows. □

For treewidth, triangulations appeared to be a very convenient and powerful tool. One of the main properties of triangulations for treewidth which was heavily exploited in numerous algorithms is the following fact: for any graph G there is a *minimal* triangulation H of G, such that $\mathrm{tw}(G) = \mathrm{tw}(H)$. By Lemma 1, there is a similar relation for branch decomposition. However, and here is the first big difference to treewidth, optimal branch decompositions may never lead to minimal triangulations.

Proposition 2. *Let K_9^- be the graph obtained from the complete graph on 9 vertices by removing a unique edge $\{a, b\}$. For every optimal branch-decomposition (T, τ) of K_9^-, $H(T, \tau)$ is not a minimal triangulation of K_9^-.*

Proof. We show that for every optimal branch-decomposition (T, τ) of K_9^-, the vertices a and b are adjacent in $H(T, \tau)$. Since K_9^- is chordal, this would imply that $H(T, \tau)$ is not a minimal triangulation of K_9^-.

Since the branchwidth of a graph on n vertices is at most $\lceil 2n/3 \rceil$ (see [27]), we have that $\mathrm{bw}(K_9^-) \leq 6$. Let (T, τ) be an optimal decomposition of K_9^-. Assume that the vertices a and b are not adjacent in $H(T, \tau)$. Then there is a branch e of T separating, in T, the sub-trees T_a and T_b. By Lemma 1, every path joining a and b in $H(T, \tau)$ intersects the vertex subset $\mathrm{mid}(e)$. Therefore $\mathrm{mid}(e)$ separates a and b in the graph G. There are at least 7 vertex disjoint paths connecting a and b in K_9^-. By Menger's theorem, $\mathrm{mid}(e)$ has at least 7 vertices, contradicting the fact that the width of (T, τ) is at most 6. □

Since optimal branch decompositions do not necessarily lead to minimal triangulations, many existing tools that make use of minimal triangulations can not be applied on branchwidth. The second important difference with treewidth is that the branchwidth problem remains NP-hard, even for a restricted class of chordal graphs, the *split* graphs [21] — for which the treewidth problem is trivial. 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 triangulations, defined in the next section, behave somehow similar to minimal triangulations.

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. A *connected component* C of $G - S$ is a set of vertices such that $G[C]$ is a maximal connected subgraph of $G - S$. 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 3. 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 $G - S$.

Efficient triangulations were introduced in [4] (actually the authors used to call them "minimal triangulations"). In particular, all the minimal triangulations of G are efficient [24].

Definition 4. A (possibly relaxed) branch decomposition (T, τ) of $G = (V, E)$ *respects* a set of vertices $S \subseteq V$ if there is a branch e of T such that $S \subseteq \mathrm{mid}(e)$.

Author's personal copy

One of the main ingredients of our result is the following result of Mazoit.

Proposition 5 ([22,23]). *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, (T, τ) respects each minimal separator of H.*

Before giving the next definition, let us provide some intuition. We want to define a structure in a graph G that corresponds to a maximal clique in some efficient triangulation of G. The following properties of maximal cliques in chordal graph are important for our purposes.

Proposition 6 ([5]). *Let H be a chordal graph and Ω be a maximal clique of H. Then for each connected component C_i of $H - \Omega$,*

- *the neighborhood $S_i = N(C_i)$ is a minimal separator;*
- *$\Omega \setminus S_i$ is non empty and is contained in a connected component of $H - S_i$.*

Definition 7. 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 is 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 minimal separators that border B.

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

By Proposition 6, every maximal clique of a chordal graph is a block of H. We prove that if H is an efficient triangulation of G, then every maximal clique Ω of H is a block of G.

Lemma 8. *Let H be an efficient triangulation of G. Then every maximal clique Ω of H is a block of G. Conversely, for each block B of G, there is an efficient triangulation $H(B)$ of G such that B is a maximal clique in H.*

Proof. Let H be an efficient triangulation of G. By Proposition 6 every maximal clique Ω of H is a block. By definition of efficient triangulations, a block of H is also a block of G.

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

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

The minimum can be taken over all minimal triangulations H of G. A similar formula can be obtained for branchwidth (see Theorem 13 or [22]).

Definition 9 (*Block-Branchwidth*). Let B be a block of G and $K(B)$ be the complete graph with vertex set B. A relaxed branch decomposition of $K(B)$ respecting each minimal separator $S \in \mathcal{S}(B)$ is called a *block-branch decomposition* of the block B. The *block-branchwidth* $\mathrm{bbw}(B)$ of B is the minimum width over all the block-branch decompositions of 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 will allow us to express the branchwidth of G by a formula similar to Eq. (1).

Before stating the main theorem of this section, let us make some easy observations.

Definition 10. Let $G = (V, E)$ be a graph and G_1 and G_2 be two sub-graphs of G such that $G = G_1 \cup G_2$. Let (T_1, τ_1) and (T_2, τ_2) be relaxed branch decompositions of G_1 and G_2. We say that a relaxed branch-decomposition (T, τ) of G is obtained by *gluing* a branch e_1 of T_1 and e_2 of T_2, if (T, τ) is obtained from (T_1, τ_1) and (T_2, τ_2) by putting a vertex v_i of degree two on e_i, $i = 1, 2$, and by adding a new edge $\{v_1, v_2\}$.

The proof of the next lemma follows from the definition of gluing.

Lemma 11. *Let (T, τ) be the relaxed branch-decomposition obtained by gluing the branches e_1 and e_2 of decompositions (T_1, τ_1) and (T_2, τ_2). Suppose that one of the following holds:*

- $\mathrm{mid}_{(T_1,\tau_1)}(e_1) \subseteq \mathrm{mid}_{(T_2,\tau_2)}(e_2)$, *or*
- $\mathrm{mid}_{(T_1,\tau_1)}(e_1) \cap \mathrm{mid}_{(T_2,\tau_2)}(e_2) = V_1 \cap V_2$.

Then $\mathrm{width}(T, \tau) = \max\{\mathrm{width}(T_1, \tau_1), \mathrm{width}(T_2, \tau_2)\}$. *Moreover, for each branch of T_1 (resp. of T_2), its middle set in (T, τ) is the same as in (T_1, τ_1) (resp. (T_2, τ_2)).*

Author's personal copy

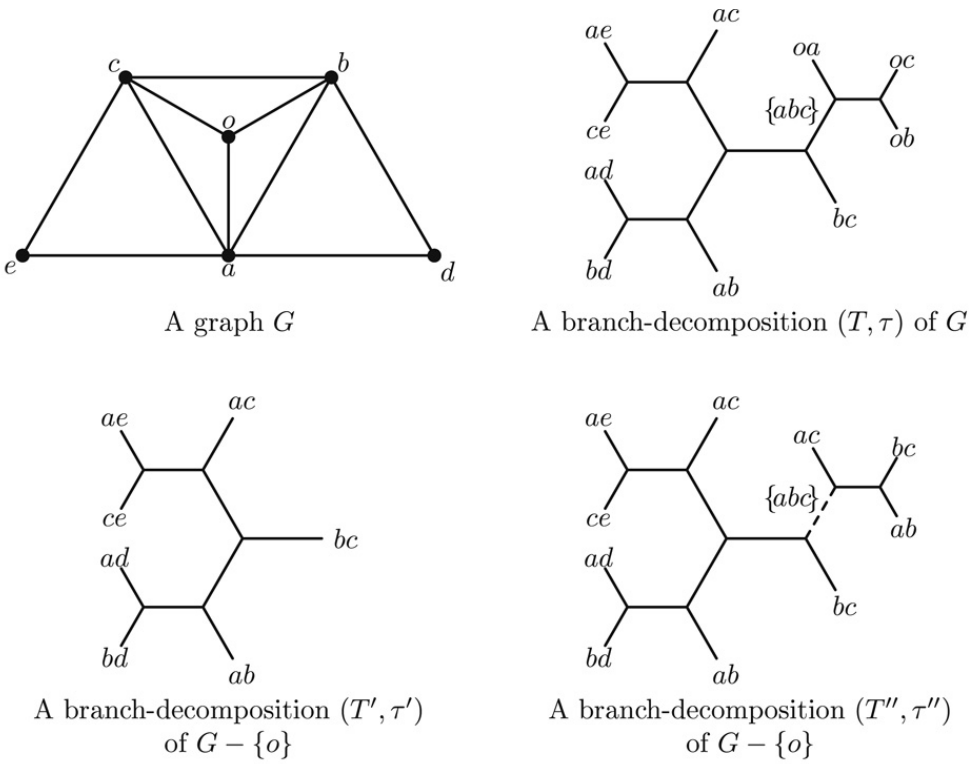


Fig. 1. Branch decomposition of $G[W]$ respecting the set S_i.

Lemma 12. *Let (T, τ) be a relaxed branch decomposition of a graph $G = (V, E)$ and let $S_1, S_2, \ldots, S_p$ be a set of cliques of G, such that (T, τ) respects each S_i. Let $W \subseteq V$ be a set of vertices containing $S_1, S_2, \ldots, S_p$. Then there is a relaxed branch decomposition (T', τ') of $G[W]$, such that*

- *(T', τ') respects each set S_i;*
- *$\text{width}(T', \tau') \leq \text{width}(T, \tau)$.*

Proof. A relaxed branch decomposition of $G[W]$ can be obtained from (T, τ) by removing edges that are not in $G[W]$. However, this decomposition does not necessary respect all sets S_i. So we apply the following trick (see Fig. 1).

For each S_i, let (T_i, τ_i) be an arbitrary branch decomposition of the clique $K(S_i)$ with vertex set S_i. We glue this decomposition to T on the branch e_i of T which respects S_i. That is, we add a node on e_i and a node on some branch of T_i and make them adjacent. We denote this new edge by e'_i. The middle set of e'_i is exactly S_i. In this way we obtain a relaxed branch decomposition (T'', τ'') of G of the same width as (T, τ). By removing from T'' all the leaves that do not correspond to edges in $G[W]$, we obtain a relaxed branch decomposition (T', τ') of $G[W]$. Every edge $\{x, y\}$ of $G[S_i]$ is mapped on some leaf of T and on some leaf of T_i, thus every vertex of S_i is in the middle set of e'_i in the relaxed branch decomposition (T', τ'). Thus (T', τ') respects all sets S_i. By Lemma 11, $\text{width}(T', \tau') \leq \text{width}(T, \tau)$. □

The following result is taken from Mazoit's Ph.D. thesis, we provide the proof here for completeness.

Theorem 13 ([22]).

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

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 Proposition 5. We prove that $\text{bbw}(\Omega) \leq \text{bw}(G)$ for each maximal clique Ω of H. Construct, like in Lemma 1 a branch decomposition (T', τ') of H having the same width as (T, τ). Let Ω be a maximal clique of G. By Lemma 8, Ω is a block of G and by Proposition 5 each minimal separator bordering Ω is contained in the middle set of some branch of T, and thus of T'. By Lemma 12, there is a relaxed branch decomposition (T'', τ'') of $G[\Omega]$ respecting each minimal separator on the border of Ω, which is a block-branch decomposition of Ω.

Conversely, let H be an efficient triangulation of G. We claim that $\text{bw}(G) \leq \max\{\text{bbw}(\Omega) \mid \Omega \text{ maximal clique of } H\}$. For each maximal clique Ω of G, let (T_Ω, τ_Ω) be an optimal block-branch decomposition of the block Ω. We glue these

Author's personal copy

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. [17]). A clique tree is defined by a tree $T = (V_T, E_T)$ and a one-to-one mapping 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 $u_{\Omega'}$ in T (u_Ω and $u_{\Omega'}$ denote the nodes associated with 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' [17]. Let e_S (resp. e'_S) be a branch of T_Ω (resp. $T_{\Omega'}$) whose middle set contains S. We obtain a relaxed branch decomposition of H by gluing the branches e_S and e'_S, for all branches $\{u_\Omega, u_{\Omega'}\}$ of T. By the properties of the clique tree, the middle set of each newly created edge connecting T_Ω and $T_{\Omega'}$ is exactly $S = \Omega \cap \Omega'$. Consequently, the middle sets of the branches contained in some T_Ω do not change. Hence $\mathrm{bw}(H) \leq \max\{\mathrm{bbw}(\Omega) \mid \Omega \text{ maximal clique of } H\}$. Since G is a sub-graph of H, we have that $\mathrm{bw}(G) \leq \mathrm{bw}(H)$ and Theorem follows. □

4. Computing the branchwidth from the block-branchwidth

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 [5]. Using the Eq. (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 [13], where it is shown that for a given a graph G and the set Π_G of its potential maximal cliques, there is an algorithm computing the treewidth of G in $\mathcal{O}(n^3|\Pi_G|)$ time.

According to Lemma 8, 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 [13].

Using Eq. (2) instead of Eq. (1) and blocks instead of potential maximal cliques, we transform the algorithm from [13] 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 $\mathcal{O}(n^3|\mathcal{B}_G|)$ time.

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

Definition 14. We denote by $\mathrm{bw}^+(S, G)$, the minimum width over all relaxed branch decompositions of G respecting S.

Our algorithm computes $\mathrm{bw}^+(S, R(S, C))$, for each pair (S, C) where S is a minimal separator of G and C is a connected component of $G - S$.

The result claimed in the following lemma is similar to Corollary 4.5 in [5].

Lemma 15. *For any graph G,*

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

Proof. Clearly $\mathrm{bw}(G) \leq \lceil 2n/3\rceil$. To prove

$$\mathrm{bw}(G) \leq \min_{S\in\Delta_G}\max_{C\in\mathcal{C}(S)} \mathrm{bw}^+(S, R(S, C)),$$

let S be a minimal separator of G. For each connected component C_i of $G - S$, let (T_{C_i}, τ_{C_i}) a relaxed branch decomposition of $R(S, C_i)$, such that S is contained in the middle set of some branch e_{C_i}. Note that G is the union of the subgraphs $R(S, C_1), \ldots, R(S, C_p)$. By iteratively gluing the branch-decompositions (T_{C_i}, τ_{C_i}) along the branches e_{C_i}, we obtain a relaxed branch-decomposition (T, τ) of G. Moreover, since for every i and j, $\mathrm{mid}(e_{C_i}) \cap \mathrm{mid}(e_{C_j}) = S = V(R(S, C_i)) \cap V\left(R(S, C_j)\right)$. By Lemma 11, the width of (T, τ) is at most the maximum width of (T_{C_i}, τ_{C_i}), over all components C_i of $G - S$, hence $\mathrm{bw}(G)$ is at most this maximum.

Conversely, let (T, τ) be an optimal branch decomposition of G inducing an efficient triangulation $H(T, \tau)$. If $H(T, \tau)$ is the complete graph, then $\mathrm{bw}(G) = \mathrm{bw}(H(T, \tau)) = \lceil 2n/3\rceil$. Otherwise let S be a minimal separator of H. By the definition of an efficient triangulation, S is also a minimal separator of G and every connected component C of $H - S$ is also a component of $G - S$. We show that $\mathrm{bw}^+(S, R(S, C)) \leq \mathrm{bw}(G)$. By Proposition 5, S is in the middle set of some branch of T. Recall that S is a clique in $H(T, \tau)$. Let H' be the subgraph of $H(T, \tau)$ induced by $S \cup C$. By Lemma 12, H' has a relaxed branch decomposition (T', τ') of width at most $\mathrm{bw}(G)$, such that S is contained in the middle set of some branch of T'. Since $R(S, C)$ is a subgraph of H', we have that by Lemma 12, there is a relaxed branch decomposition of $R(S, C)$ respecting S, of width at most $\mathrm{width}(T', \tau')$. Thus $\mathrm{bw}^+(S, R(S, C)) \leq \mathrm{width}(T', \tau') \leq \mathrm{bw}(G)$. We conclude that $\mathrm{bw}(G) \geq \max_{C\in\mathcal{C}(S)} \mathrm{bw}^+(S, R(S, C))$. □

The next result is the analogue of Corollary 4.5 in [5].

Lemma 16. *Let S be a minimal separator of G and C be a component of $G - S$. Suppose that $S' = N(C)$ is strictly contained in S. Then*

$$\mathrm{bw}^+(S, R(S, C)) = \max\left\{|S|, \mathrm{bw}^+\left(S', R(S', C)\right)\right\}.$$

Author's personal copy

Proof. Note that since $R(S', C)$ is a subgraph of $R(S, C)$, by Lemma 12, $\mathrm{bw}^+\left(S', R(S', C)\right) \leq \mathrm{bw}^+(S, R(S, C))$. By definition, $\mathrm{bw}^+(S, R(S, C)) \geq |S|$, hence

$$\mathrm{bw}^+(S, R(S, C)) \geq \max\left\{|S|, \mathrm{bw}^+\left(S', R(S', C)\right)\right\}.$$

Let (T', τ') be a relaxed branch decomposition of $R(S', C)$, with S' being contained in the middle set of a branch e'_S. Take an optimal relaxed branch decomposition (T_S, τ_S) of the clique $K(S)$. Let e_s be an branch of (T_S, τ_S). By gluing (T', τ') and (T, τ) on the branches e_S and e'_S, we obtain the relaxed branch decomposition of $R(S', C)$ respecting S. By Lemma 11, the width of this decomposition does not exceed $\max\{|S|, \mathrm{bw}^+(S', R(S', C))\}$. □

The following lemma from [5] uses minimal triangulations and potential maximal cliques but despite that, its proof also holds for efficient triangulations and blocks.

Lemma 17 (*Lemma 4.6 in [5]*). *Let S be a minimal separator of a graph G and let C be a connected component of $G - S$ such that $N(C) = S$. For every efficient triangulation $H(S, C)$ of $R(S, C)$, there is a maximal clique Ω of $H(S, C)$ such that $S \subset \Omega$ and Ω is a block of G.*

The following lemma is similar to Corollary 4.8 in [5].

Lemma 18. *Let S be a minimal separator of G and C be a component of $G - S$ such that $S = N(C)$. Then*

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

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

Proof. Let Ω be a block of G such that $S \subset \Omega \subseteq S \cup C$. Consider an optimal block-branch decomposition (T_Ω, τ_Ω) of Ω and for each minimal separator S_i, let f_i be branch of T_Ω whose middle set contains S_i. Let (T_i, τ_i) an optimal relaxed branch decomposition of $R(S_i, C_i)$ respecting S_i. Denote by e_i the branch of T_i whose middle set contains S_i. We glue (T_Ω, τ_Ω) to each (T_i, τ_i) on the branches f_i and e_i. Hence we obtain a relaxed branch decomposition of $R(S, C)$ respecting S, of width at most $\max(\mathrm{bbw}(\Omega), \mathrm{bw}(R(S_i, C_i)))$, over all components C_i.

Conversely, we prove first that there is an optimal relaxed branch decomposition (T, τ) of $R(S, C)$, respecting S, such that $H(T, \tau)$ is an efficient triangulation of $R(S, C)$. Let $k = \mathrm{bw}^+(S, R(S, C))$ (in particular $k \geq |S|$). Consider a complete graph $K_d = (V_d, E_d)$ with $\lfloor 3k/2 \rfloor$ vertices such that $S \subseteq V_d$ and $V_d \cap C = \emptyset$. Let G' be the union of K_d and $R(S, C)$. We have that $\mathrm{bw}(G') = k$. Clearly $\mathrm{bw}(G') \geq \mathrm{bw}(K_d) \geq k$. To prove that $\mathrm{bw}(G') = k$, it is sufficient to find a relaxed branch decomposition of K_d respecting S, because then this decomposition can be glued with an optimal relaxed branch decomposition of $R(S, C)$ respecting S. So let A, B, C be three subsets of V_d of size at most k, such that $S \subseteq A$ and every vertex of V_d is in exactly two of the subsets (hence each edge of K_d has both endpoints in one of the sets). These sets exist by the fact that $|V_d| = \lfloor 3k/2 \rfloor$. Consider three branch decompositions corresponding to the complete graphs $K_d[A]$, $K_d[B]$ and $K_d[C]$. Add a new node u and, for each of the three decompositions, make u adjacent to the middle of some branch. Note that every middle set of the new decomposition is contained in A, B or C. Also the branch linking u to the decomposition of $G'[A]$ has middle set $A \cap (B \cup C) = A$, so this relaxed branch decomposition respects S and $\mathrm{bw}(G') = k$ as claimed.

Let now (T', τ') be an optimal branch decomposition of G', such that $H(T', \tau')$ is an efficient triangulation of G' (see Proposition 5). We claim that S is a minimal separator of $H(T', \tau')$ and therefore of G'. Note that V_d is a maximal clique in $H(T', \tau')$. (Otherwise $H(T', \tau')$ has a clique of size strictly larger than $\lfloor 3k/2 \rfloor$, contradicting the fact that its branchwidth is k.) Let C' be the component of $H(T', \tau') - V_d$ containing C and let S' be the neighborhood of C' in $H(T', \tau')$. Since S' is a minimal separator of the efficient triangulation $H(T', \tau')$, S' is also a minimal separator of G'. By construction, the only minimal separator of G' contained in V_d is exactly S. Then the subgraph H_R induced by $S \cup C$ in $H(T', \tau')$ is an efficient triangulation of $R(S, C)$. Moreover, by Proposition 5, (T', τ') respects S. By Lemma 12, we can construct from (T', τ') a relaxed branch decomposition (T, τ) respecting S and such that $H(T, \tau) = H_R$. Hence $H(T, \tau)$ is an efficient triangulation of $R(S, C)$.

By Lemma 17, there is a maximal clique Ω of $H(T, \tau)$ such that $S \subset \Omega \subseteq S \cup C$. By Lemma 8, Ω is a block of G. By restricting, like in Lemma 12, the relaxed branch decomposition (T, τ) to each of the graphs $R(S_i, C_i)$ we deduce that $\mathrm{bw}^+(S, R(S, C)) \leq \mathrm{width}(T, \tau)$. Similarly we restrict (T, τ) to a block-branch decomposition of Ω and conclude that $\mathrm{bbw}(\Omega) \leq \mathrm{width}(T, \tau)$. □

The algorithm for computing the branchwidth of G is shown in Fig. 2. It can be seen as the translation of the algorithm from [13], to efficient triangulations and blocks by making use of Lemmas 15–18.

Theorem 19. *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 first `for` loop of the algorithm simply applies Lemmas 16 and 18, in order to compute $\mathrm{bw}^+(S, R(S, C))$ for each minimal separator S of G and each component C of $G - S$. We only need to notice that, in each of these lemmata, in order to compute $\mathrm{bw}^+(S, R(S, C))$ we need to know the similar quantity for couples (S', C') of strictly smaller size.

Let us discuss an implementation of the algorithm running in $\mathcal{O}^*(|\mathcal{B}_G|)$ time.

Author's personal copy

Algorithm computing the branchwidth of a graph

Input: G, all its blocks and all its minimal separators
Output: $\mathrm{bw}(G)$
begin
  compute all the couples (S, C) where S is a minimal separator and
    C a component of $G - S$;
  sort them by the size of $S \cup C$
  for each (S, C) taken in increasing order
    if $S' = N(C)$ is strictly contained in S
      $\mathrm{bw}^+\big(S, R(S,C)\big) = \max\big\{|S|, \mathrm{bw}^+\big(S', R(S',C')\big)\big\}$
    else
      $\mathrm{bw}^+\big(S, R(S,C)\big) := \mathrm{bbw}(S \cup C)$
      for each block Ω with $S \subset \Omega \subseteq S \cup C$
        compute the components C_i of $G - \Omega$ contained in C and
          the sets $S_i = N(C_i)$
        $\mathrm{bw}^+\big(S, R(S,C)\big) := \min\Big\{\mathrm{bw}^+\big(S, R(S,C)\big),$
          $\max_i\big\{\mathrm{bbw}(\Omega), \mathrm{bw}^+\big(S_i, R(S_i,C_i)\big)\big\}\Big\}$
      end_for
    end_if
  end_for
  let Δ_G be the set of minimal separators of G
  $\mathrm{bw}(G) := \min\{\lceil 2n/3\rceil, \min_{S\in\Delta_G} \max_{C\in\mathcal{C}(S)} \mathrm{bw}^+\big(S, R(S,C)\big)\}$
end

Fig. 2. Algorithm computing the branchwidth of a graph.

To store and manipulate the minimal separators, blocks and couples (S, C), we use data structures that allow us to search and to insert an element in $\mathcal{O}(n)$ time.

During a preprocessing step, we realize the following operations.

- Compute the list of all couples (S, C) such that S is a minimal separator and C is a component of $G - S$. For each minimal separator S, we compute the components of $G - S$ and store a pointer towards each couple of type (S, C). Given a minimal separator S, there are at most n couples associated with it, so at most n pointers to be stored. Then, for each couple (S, C) such that $S' = N(C)$ is strictly contained into S, we store a pointer from (S, C) to (S', C'). The last operation requires $\mathcal{O}(n)$ time, that is the time to search for (S', C) into the list of couples. Hence the whole step costs $\mathcal{O}(m|\Delta_G|)$ time.
- For each block Ω, compute the components C_i of $G - \Omega$ and then store a pointer from Ω to the couple $(N(C_i), C_i)$. There are at most n such blocks. This computation can be done in $\mathcal{O}(n^2)$ time for each block, so globally in $\mathcal{O}(n^2|\mathcal{B}_G|)$ time.
- Compute all the *good triples* (S, C, Ω), where (S, C) is a couple with $S = N(C)$ and Ω is a block such that $S \subset \Omega \subseteq S \cup C$. Moreover, for each good triple we store a pointer from (S, C) to Ω. Note that $S \in \mathcal{S}(\Omega)$ and, by definition of a block, there are at most n minimal separators $S \subset \Omega$ in its border. For each such S there is exactly one component $G - S$ intersecting Ω (in particular there are at most $n|\mathcal{B}_G|$ good triples). For each component C' of $G - \Omega$ we take $S = N(C')$, find the component C of $G - S$ intersecting Ω and store the pointer from (S, C) to Ω. Thus this computation takes $\mathcal{O}(nm)$ time for each block, so $\mathcal{O}(nm|\mathcal{B}_G|)$ globally.

Hence this preprocessing step costs $\mathcal{O}(m|\Delta_G| + nm|\mathcal{B}_G|)$. Sorting the couples (S, C) 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 $\mathcal{O}(m)$, for the components C_i, and the sets N_i can be computed by a graph search and by the fact that there are at most n couples (S_i, C_i) associated to a block. With the data structures obtained during the preprocessing step, each couple (S, C) keeps a pointer towards each block Ω, such that (S, C, Ω) form a good triple. Thus the number of iterations of the two nested loops is exactly the number of good triples, that is at most $n|\mathcal{B}_G|$. It follows that the two loops cost $\mathcal{O}(nm|\mathcal{B}_G|)$ time.

Each minimal separator S keeps the list of couples of type (S, C), 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 $\mathcal{O}(m|\Delta_G| + nm|\mathcal{B}_G|)$. Each minimal separator is contained in at least one block. According to their definition, each block contains at most n minimal separators. Each minimal separator is contained in at least one block, therefore $|\mathcal{B}_G| \geq |\Delta_G|/n$. We conclude that the algorithm runs in $\mathcal{O}(nm|\mathcal{B}_G|)$ time.

Author's personal copy

The algorithm can be transformed in order to output not only the branchwidth of the graph, but also an optimal branch decomposition. □

5. 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 can be deduced from [21].

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 $n(B) + s(B) \leq n$.

Let us mention that a result very similar to our Lemma 20 has been independently given in [26]. The authors call their decomposition a *troika*.

Lemma 20. *For any block B of a graph G, $\mathrm{bbw}(B) \leq p$ if and only if there is a partition of B into four parts A_1, A_2, A_3, D such that*

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

Proof. Suppose that $\mathrm{bbw}(B) \leq p$, and let (T, τ) be an optimal block-branch decomposition of B. Recall that this relaxed 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. For every $x, y \in B$, the sub-trees T_x and T_y share a vertex. By the Helly property, there is a node u of T contained in all the sub-trees of the form T_x. Clearly u is a ternary node, except for the trivial case when $n(B) \leq 2$. 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 $i \in \{1, 2, 3\}$. We set

$$B_i = \{z \in B \mid z \text{ is incident to some edge of } K(B) \text{ mapped on a leaf of } T(i)\}$$

and for each triple (i, j, k) with $i, j, k \in \{1, 2, 3\}$ and $i \neq j \neq k$, we put

$$D = B_1 \cap B_2 \cap B_3, \quad \text{and} \quad A_i = B_j \cap B_k \setminus D.$$

Observe that D, A_1, A_2, A_3 form a partition of B. In fact, 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. We claim that for every $i \in \{1, 2, 3\}$,

$$\mathrm{mid}(e_i) = B_i = B \setminus A_i. \tag{3}$$

In fact,

$$\mathrm{mid}(e_i) = (B_i \cap B_j) \cup (B_i \cap B_k) = A_j \cup A_k \cup D = B \setminus A_i.$$

By (3), $|B \setminus A_i| = \mathrm{mid}(e_i) \leq p$, and the first condition of the lemma holds. To prove the second condition, let us consider a separator $S \in \mathcal{S}(B)$. By the definition of block-branch decomposition, S is contained in B_i for some $i \in \{1, 2, 3\}$. By (3), $S \subseteq B \setminus A_i$.

To prove the "only if" part, let us assume that there is a partition satisfying the two conditions of the lemma. We construct a block-branch decomposition of 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 of degree two 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 $\mathrm{mid}(e_i) = B_i \cap (B_j \cup B_k)$, where $\{i, j, k\} = \{1, 2, 3\}$. Since

$$B_i = B \setminus A_i \subseteq (B \setminus A_j) \cup (B \setminus A_k) = B_j \cup B_k,$$

we have that $\mathrm{mid}(e_i) = B_i$. Consequently, the relaxed branch decomposition respects the minimal separators on the border of B. Clearly for each branch e of T, $\mathrm{mid}(e)$ is contained in some B_i, so $|\mathrm{mid}(e)| \leq p$ and Lemma follows. □

Lemma 21. *The block-branchwidth of a block B can be computed in $\mathcal{O}^*(3^{s(B)})$ time.*

Proof. Let B be a block of G. Suppose that $\mathrm{bbw}(B) \leq p$. By Lemma 20, there exists a partition A_1, A_2, A_3 and D of B such that $|B \setminus A_i| \leq p$ 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) $a_i \geq 0, d \geq 0, a_1 + a_2 + a_3 + d = n(B)$;
(2) $|S_1 \cap S_2 \cap S_3| \leq d, |(S_1 \cap S_2) \setminus S_3| \leq a_3,$
$|(S_2 \cap S_3) \setminus S_1| \leq a_1, |(S_3 \cap S_1) \setminus S_2| \leq a_2$;
(3) $a_1 + a_2 + d \leq p, a_2 + a_3 + d \leq p, a_3 + a_1 + d \leq p$.

Author's personal copy

The first inequalities 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 $\text{bbw}(B) \leq p$.

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 $|B \setminus A_i| \leq p$ by the third series of inequalities, so by Lemma 20 we have $\text{bbw}(B) \leq p$.

Hence, there is a block-branch decomposition of B of branchwidth at most p if and only if there is a 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 $\text{bbw}(B) \leq p$ 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. □

Lemma 22. *The block-branchwidth of a 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 20 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 would bring us to time $\mathcal{O}^*(3^{n(B)})$ algorithm.

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 [8]). 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. □

Since at least one of $s(B)$ or $n(B)$ is at most half of the vertices of the graph, Lemmas 21 and 22 imply the following theorem.

Theorem 23. *For any block B of a graph G on n vertices, the block-branchwidth of B can be computed in $\mathcal{O}^*(\sqrt{3}^n)$ time.*

Theorems 19 and 23 imply our main result.

Theorem 24. *The branchwidth of graph on n vertices can be computed in $\mathcal{O}^*((2\sqrt{3})^n)$ time and $\mathcal{O}^*(2^n)$ space.*

Proof. The algorithm tries all vertex subsets B of a graph G and checks if B is a block of G. Clearly, we can verify if B is a block in polynomial time. If B is a block, we compute the block branchwidth of B making use of Theorem 23. 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 $\mathcal{O}^*(2^n)$.

We use Theorem 19 for computing the branchwidth of G. The second phase takes $\mathcal{O}^*(2^n)$ time and space. □

6. 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 $O(\min(3^{n'}, 3^{s'}))$-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.

Acknowledgements

The first author has an additional support by the Research Council of Norway and the second author has an additional support by the Université de Provence.

Author's personal copy

References

[1] Michael Alekhnovich, Alexander A. Razborov, Satisfiability, branch-width and Tseitin tautologies, in: Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, FOCS'02, 2002, pp. 593–603.
[2] Richard Beigel, David Eppstein, 3-coloring in time $O(1.3289^n)$, Journal of Algorithms 54 (2) (2005) 168–204.
[3] Hans L. Bodlaender, A partial k-arboretum of graphs with bounded treewidth, Theoretical Computer Science 209 (1998) 1–45.
[4] Hans L. Bodlaender, Ton Kloks, Dieter Kratsch, Treewidth and pathwidth of permutation graphs, SIAM Journal on Discrete Mathematics 8 (1995) 606–616.
[5] Vincent Bouchitté, Ioan Todinca, Treewidth and minimum fill-in: Grouping the minimal separators, SIAM Journal on Computing 31 (2001) 212–232.
[6] Jesper Makholm Byskov, Enumerating maximal independent sets with applications to graph colouring, Operations Research Letters 32 (2004) 547–556.
[7] William John Cook, Paul D. Seymour, Tour merging wia branch-decomposition, INFORMS Journal on Computing 15 (2003) 233–248.
[8] Thomas H. Cormen, Charles Eric Leiserson, Ronald Linn Rivest, Introduction to Algorithms, MIT Press, 1990.
[9] Evgeny Dantsin, Andreas Goerdt, Edward A. Hirsch, Ravi Kannan, Jon Michael Kleinberg, Christos H. Papadimitriou, Prabhakar Raghavan, Uwe Schning, A deterministic $(2 - 2/(k + 1))^n$ algorithm for k-SAT based on local search, Theoretical Computer Science 289 (1) (2002) 69–83.
[10] Erik D. Demaine, Fedor V. Fomin, MohammadTaghi Hajiaghayi, Dimitrios M. Thilikos, Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs, ACM Transactions on Algorithms 1 (1) (2005) 33–47.
[11] Frederic Dorn, Eelko Penninkx, Hans L. Bodlaender, Fedor V. Fomin, Efficient exact algorithms on planar graphs: Exploiting sphere cut branch decompositions, in: Proceedings of the 13rd Annual European Symposium on Algorithms, ESA'05, in: Lecture Notes in Computer Science, vol. 3669, 2005, pp. 95–106.
[12] Fedor V. Fomin, Fabrizio Grandoni, Dieter Kratsch, Measure and conquer: A simple $O(2^{0.288n})$ independent set algorithm, in: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'06, 2006, pp. 18–25.
[13] Fedor V. Fomin, Dieter Kratsch, Ioan Todinca, Exact (exponential) algorithms for treewidth and minimum fill-in, in: Proceedings of the 31th International Colloquium on Automata, Languages, and Programming, ICALP'04, in: Lecture Notes in Computer Science, vol. 3142, 2004, pp. 568–580.
[14] Fedor V. Fomin, Dimitrios M. Thilikos, Dominating sets in planar graphs: Branchwidth and exponential speed-up, SIAM Journal of Computing 36 (2) (2006) 281–309.
[15] Fedor V. Fomin, Dimitrios M. Thilikos, New upper bounds on the decomposability of planar graphs, Journal of Graph Theory 51 (1) (2006) 53–81.
[16] Fanica Gavril, The intersection graphs of a path in a tree are exactly the chordal graphs, Journal of Combinatorial Theory Series B 16 (1974) 47–56.
[17] Martin Charles Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, 1980.
[18] Qian-Ping Gu, Hisao Tamaki, Optimal branch-decomposition of planar graphs in $O(n^3)$ time, in: Proceedings of the 32th International Colloquium on Automata, Languages and Programming, ICALP'05, in: Lecture Notes in Computer Science, vol. 3580, 2005, pp. 373–384.
[19] Kazuo Iwama, Suguru Tamaki, Improved upper bounds for 3-SAT, in: Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'04, 2004, pp. 823–832.
[20] Tang Jian, An $O(2^{0.304n})$ algorithm for solving maximum independent set problem, IEEE Transactions on Computers 35 (9) (1986) 847–851.
[21] Ton Kloks, Jan Kratochvíl, Haiko Müller, Computing the branchwidth of interval graphs, Discrete Applied Mathematics 145 (2) (2005) 266–275.
[22] Frédéric Mazoit, Décomposition algorithmique des graphes, Ph.D. Thesis, École normale supérieure de Lyon, 2004 (in French).
[23] Frédéric Mazoit, The branch-width of circular-arc graphs, in: Proceedings of the 7th Latin American Theoretical Informatics Symposium, LATIN'06, in: Lecture Notes in Computer Science, vol. 3887, 2006, pp. 727–736.
[24] Andreas Parra, Petra Scheffler, Characterizations and algorithmic applications of chordal graph embeddings, Discrete Applied Mathematics 79 (1–3) (1997) 171–188.
[25] Ramamohan Paturi, Pavel Pudlk, Michael E. Saks, Francis Zane, An improved exponential-time algorithm for k-SAT, in: Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science, FOCS'98, 1998, pp. 628–637.
[26] Christophe Paul, Jan Arne Telle, New tools and simpler algorithms for branchwidth, in: Proceedings of the 13rd Annual European Symposium on Algorithms, ESA'05, in: Lecture Notes in Computer Science, vol. 3669, 2005, pp. 379–390.
[27] Neil Robertson, Paul D. Seymour, Graph minors. X. Obstructions to tree-decomposition, Journal of Combinatorial Theory Series B 52 (2) (1991) 153–190.
[28] Mike Robson, Algorithms for maximum independent sets, Journal of Algorithms 7 (3) (1986) 425–440.
[29] Uwe Schoning, A probabilistic algorithm for k-SAT and constraint satisfaction problems, in: Proceedings of the 40th annual IEEE symposium on foundations of computer science, FOCS'99, 1999, pp. 410–414.
[30] Paul D. Seymour, Robin Thomas, Call routing and the ratcatcher, Combinatorica 14 (2) (1994) 217–241.
[31] Yngve Villanger, Improved exponential-time algorithms for treewidth and minimum fill-in, in: Proceedings of the 7th Latin American Theoretical Informatics Symposium, LATIN'06, in: Lecture Notes in Computer Science, vol. 3887, 2006, pp. 800–811.
[32] Ryan Williams, A new algorithm for optimal 2-constraint satisfaction and its implications, Theoretical Computer Science 348 (2–3) (2005) 357–365.
[33] Gerhard J. Woeginger, Exact algorithms for NP-hard problems: A survey, in: Proceedings of the 5th International Workshop on Combinatorial Optimization — Eureka, You Shrink!, in: Lecture Notes in Computer Science, vol. 2570, 2003, pp. 185–207.