Journal of Algorithms **32,** 167–194 (1999)
Article ID jagm.1999.1011, available online at http://www.idealibrary.com on IDEAL®

Graphs with Branchwidth at Most Three*

Hans L. Bodlaender

Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, the Netherlands
E-mail: hansb@cs.uu.nl

and

Dimitrios M. Thilikos†

Department of Computer Science DC 2117, University of Waterloo, 200 University Avenue West, Waterloo, Ontario N2L 32G1, Canada
E-mail: sedthilk@plg.uwaterloo.ca

Received December 20, 1997; revised December 28, 1998; accepted February 9, 1999

In this paper we investigate both the structure of graphs with branchwidth at most three, as well as algorithms to recognise such graphs. We show that a graph has branchwidth at most three if and only if it has treewidth at most three and does not contain the three-dimensional binary cube graph as a minor. A set of four graphs is shown to be the obstruction set for the class of graphs with branchwidth at most three. Moreover, we give a safe and complete set of reduction rules for the graphs with branchwidth at most three. Using this set, a linear time algorithm is given that verifies if a given graph has branchwidth at most three, and, if so, outputs a minimum width branch decomposition.

Key Words: graph algorithms; branchwidth; obstruction set; graph minors; reduction rule.

* This paper is the full version of part of the paper titled "Constructive Linear Time Algorithms for Branchwidth" which appeared in the "Proceedings of ICALP'97" (see [7]). The research was partially supported by ESPRIT Long Term Research Project 20244 (Project ALCOM IT: *Algorithms and Complexity in Information Technology*).

† This research has been done while the second author was working at the Department of Computer Science of Utrecht University, Netherlands, supported by the Training and Mobility of Researchers (TMR) Program, (EU Contract ERBFMBICT950198).

0196-6774/99 $30.00

1. INTRODUCTION

This paper studies the graphs with branchwidth at most three. The notion of branchwidth has a close relationship to the more well-known notion of treewidth, a notion that has come to play a large role in many recent investigations in algorithmic graph theory. (See Section 2 for definitions of treewidth and branchwidth). One reason for the interest in this notion is that many graph problems can be solved by linear time algorithms, when the inputs are restricted to graphs with some uniform upper bound on their treewidth. Most of these algorithms first try to find a tree decomposition of small width and then utilise the advantages of the tree structure of the decomposition.

The branchwidth of a graph differs from its treewidth by at most a multiplicative constant factor (see Theorem 1.b) As branchwidth also reflects some optimal tree structure arrangement, it is possible to have algorithmic applications analogous to those of treewidth. Hence, instead of using tree decompositions, one also can use branch decompositions as the starting point for linear time algorithms for problems restricted to graphs with bounded treewidth (and, hence, also bounded branchwidth). In fact, in some cases, it appears that branchwidth is more convenient to use and seems to give better constant factors in the implementation of the algorithms; for instance, Cook used branch decompositions as an important ingredient in a practical approximation algorithm for the Travelling Salesman Problem [10] and remarked that branchwidth was the more natural notion (instead of treewidth) to use for that problem [9]; where tree decompositions primarily are concerned with vertices, branch decompositions deal more with edges (in a loose sense). We also mention that the branchwidth of planar graphs can be computed in polynomial time (see [19]). As both treewidth and branchwidth are NP-complete parameters (see [1, 19]), it appears an interesting task to find algorithms solving the following problems (k is assumed to be a fixed constant):

Treewidth: Given a graph G, check if G has treewidth at most k.

Branchwidth: Given a graph G, check if G has branchwidth at most k.

The constructive version of Treewidth (Branchwidth) is, given a graph with treewidth (branchwidth) at most k, output a minimum width tree (branch) decomposition.

According to the results of Robertson and Seymour [15, 16] for any class of graphs that is closed under taking of minors there exists a finite set of graphs, its *obstruction set*, such that a graph G belongs to the class iff no

element of the obstruction set is a minor of G. It is also known that, for any k, the class of graphs, where treewidth (or branchwidth) is bounded by a fixed k, is closed under taking of minors (see also Theorem 1.a). An immediate consequence of this fact (using results of Robertson and Seymour and the algorithm from [5]) is the existence of a linear time algorithm solving Treewidth or Branchwidth. Unfortunately, in this way, we only get a nonconstructive proof of the existence of such an algorithm; but in order to construct the algorithm, we must know the corresponding obstruction set. Additionally, we would like to have an algorithm that not only decides the branchwidth, but which also constructs the corresponding branch decomposition.

Much research has been done towards the construction of linear time algorithms solving Treewidth, Branchwidth, and their constructive versions. In [5], a linear time algorithm for treewidth was constructed. As this algorithm appears to be heavily exponential on k (and thus impractical, at least without considerable optimisations in the implementation), practical ''tailor-made'' algorithms have been presented for small values of k (treewidths 1 and 2 [14, 21], treewidth 3 [3, 12, 14], treewidth 4 [17].) Also, the obstruction sets for the class of graphs with treewidths 1, 2, and 3 are known [4, 18, 21]. Recently, a linear time algorithm solving Branchwidth and its constructive version was given in [7]. Unfortunately, the algorithms in [7] appear to be impractical, similarly to the case of treewidth.

In this paper, we provide special ''tailor made'' results for the cases where $k \leq 3$. As the cases where $k \leq 2$ are trivial we focus our attention on the case $k = 3$. More specifically, for the class of graphs with branchwidth at most three, we identify the obstruction set and we give a set of safe and complete reduction rules enabling the construction of a practical linear time algorithm that checks if a graph has branchwidth at most three and, if so, outputs a minimum width branch decomposition. The obstruction set consists of the four graphs K_5, M_6, M_8, Q_3 depicted in Fig. 2 and the proof of its correctness is based on a structural lemma asserting that the graphs of branchwidth at most three are exactly the graphs that have treewidth at most three and that do not contain the three-dimensional binary cube graph (i.e. graph Q_3 of Fig. 2) as a minor.

The paper is organized as follows. In Section 2, the basic definition and preliminary results are presented. In Section 3, we give the main routine of our algorithm, along with several graph theoretic results concerning the obstruction set for the class of graphs with branchwidth at most three. In Section 4, we identify a complete and safe set of reduction rules leading to the construction of a practical linear time algorithm solving Branchwidth and its constructive version when $k \leq 3$. Finally, an overall description of the algorithm is given in Section 5.

2. DEFINITIONS AND PRELIMINARY RESULTS

We consider undirected graphs without parallel edges or self-loops. (It is easy to extend the results to graphs with parallel edges and/or self-loops.) Given a graph $G = (V, E)$ we denote its vertex set V and edge set E with $V(G)$ and $E(G)$, respectively. A *triangle* $t = \{v_1, v_2, v_3\}$ of G is a triple of vertices in $V(G)$ such that $\{\{v_1, v_2\}, \{v_2, v_3\}, \{v_1, v_3\}\} \subseteq E(G)$. For any vertex $v \in V(G)$, we define the neighborhood of v, $N_G(v)$, as the set of vertices in $V(G)$ adjacent to v. Given a set $S \subseteq V(G)$ we denote the graph induced by S. We also denote the complete graph with r vertices by K_r and by P_r we denote the graph consisting of a path on r vertices. An *r-clique* of G is a subgraph of G isomorphic to K_r. Let T be a tree and let $S_i \subseteq V(T)$, $i = 1, 2$, be two subsets of $V(T)$ such that $G[S_i]$, $i = 1, 2$, are connected and $|V_1 \cap V_2| \leq 1$. We define the *path connecting* S_1 and S_2 as the shortest path connecting some of the vertices of S_1 with some of the vertices in S_2 (notice that such a path is uniquely defined). Finally, we will assume that all the graphs we deal with are connected, as this does not influence the generality of our results. (The branchwidth of a graph equals the maximum branchwidth of its connected components.)

Given two graphs G, H, we say that H is a *minor* of G (denoted by $H \preccurlyeq G$) if H can be obtained by a series of the following operations: vertex deletions, edge deletions, and edge contractions (a contraction of an edge $\{u, v\}$ in G is the operation that replaces u and v by a new vertex whose neighbours are the vertices that were adjacent to u and/or v). Let $\mathcal{G}$ be a class of graphs. We say that $\mathcal{G}$ is *closed under taking of minors* when all the minors of any graph in $\mathcal{G}$ belong also to $\mathcal{G}$. Given a graph class $\mathcal{G}$ that is closed under taking of minors, we define the *obstruction set* for $\mathcal{G}$ as the set of minor minimal graphs that do not belong to $\mathcal{G}$. Robertson and Seymour proved (see, e.g., [15]) that any class of graphs $\mathcal{G}$ contains a finite set of minor minimal elements. According to this result, any graph class that is closed under taking of minors has a finite obstruction set.

It follows that if $\mathcal{G}$ is closed under taking of minors, then, for any graph $G, G \in \mathcal{G}$ iff there is no graph in H in the obstruction set for $\mathcal{G}$ such that $H \preccurlyeq G$.

We give now the formal definitions of treewidth and branchwidth.

A *tree decomposition* of a graph G is a pair $(\{X_i | i \in I\}, T = (I, F))$, where $\{X_i | i \in I\}$ is a collection of subsets of V and T is a tree, such that

- $\bigcup_{i \in I} X_i = V(G)$,
- for each edge $\{v, w\} \in E(G)$, there is an $i \in I$ such that $v, w \in X_i$, and
- for each $v \in V$ the set of nodes $\{i | v \in X_i\}$ induces a subtree of T.

The *width* of a tree decomposition $(\{X_i | i \in I\},\ T = (I, F))$ equals $\max_{i \in I}\{|X_i| - 1\}$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G.

A *branch decomposition* of a graph G is a pair (T, τ), where T is a tree with vertices of degree 1 or 3, and τ is a bijection from the set of leaves of T to $E(G)$. The *order* of an edge e in T is the number of vertices $v \in V(G)$ such that there are leaves t_1, t_2 in T in different components of $T(V(T), E(T) - e)$ with $\tau(t_1)$ and $\tau(t_2)$ both containing v as an endpoint.

The *width* of (T, τ) is the maximum order over all edges of T, and the *branchwidth* of G is the minimum width over all branch decompositions of G (in the case where $|E(G)| \leq 1$, we define the branchwidth to be 0; if $|E(G)| = 0$, then G has no branch decomposition; if $|E(G)| = 1$, then G has a branch decomposition consisting of a tree with one vertex—the width of this branch decomposition is considered to be 0).

Instead of bijections, we can use different types of functions τ. If τ is a surjective function that maps every leaf of T to an edge $e \in E(G)$, then we have an *amplified branch decomposition*: for each edge $e \in E(G)$ there exists at least one leaf v or T with $\tau(v) = e$.

LEMMA 1. (i) *One can construct an algorithm that, given a branch decomposition* (T, τ) *of a graph* G *with width at most* k*, outputs a branch decomposition of any subgraph* G' *of* G *with width at most* k *in* $\mathcal{O}(|V(T)|)$ *time*.

(ii) *One can construct an algorithm that, given an amplified branch decomposition* (T, τ) *of a graph* G *with width at most* k*, outputs a branch decomposition of* G *with width at most* k*, in* $\mathcal{O}(|V(T)|)$ *time*.

Proof. (i) The algorithm is as follows: Start with the branch decomposition (T, τ). Let W be the set of leaves that are mapped to edges in $E(G')$. Now, repeat the following steps until none is possible: (i) remove a leaf not in W, and (ii) remove a vertex of degree two and connects its neighbours. One easily sees that a branch decomposition of G' results of width not larger than the width of (T, τ).

(ii) Similar. Now, for each edge $e \in E(G)$, choose an arbitrary leaf v_e that is mapped to that edge by τ, and let W be the collection of those leaves. Then apply the procedure given above with this set W.

Also, one easily sees the time needed for the procedure is $\mathcal{O}(|V(T)|)$. ∎

For an example of the second statement of Lemma 1 see Figs. 11(iii), (iv). The grey vertices are those that will be removed during step (i) and the white vertices are those that will be removed during step (ii).

In what follows we denote as $\mathscr{B}_k$ ($\mathscr{T}_k$) the obstruction set for the graphs with branchwidth (treewidth) at most k.

THEOREM 1 [16]. *The following statements hold*:

a. *The class of graphs with bounded branchwidth is closed under taking of minors*.

b. $\mathit{branchwidth}(G) \leq \mathit{treewidth}(G) + 1 \leq \lfloor \frac{3}{2}\mathit{branchwidth}(G) \rfloor$.

c. *A graph G has branchwidth at most* 0 (*at most* 1), *if and only if each connected component of G contains at most one edge* (*at most one vertex of degree at least* 2).

d. $\mathscr{B}_0 = \{P_3\}$, $\mathscr{B}_1 = \{K_3, P_4\}$, *and* $\mathscr{B}_2 = \{K_4\}$.

A *reduction rule* R is a triple (H, S, f), where H is a graph, $S \subseteq V(H)$, $S \neq \emptyset$, and f: $V(H) \to \omega + 1$ is a labelling of vertices in H by ordinals (finite ones and ω), such that $\forall v \in S$: $f(v) = 0$. We say that a reduction rule $R = (H, S, f)$ *occurs* in G if H is a subgraph of G and for any $v \in V(H)$ the degree of v in $G[(V(G) - V(H)) \cup \{v\}]$ is at most $f(v)$.

The *result of applying R on G* is the graph arising from G if we remove the vertices in S and connect as a clique in G all vertices in $V(H) - S$ (i.e., add the minimum number of edges so that $V(H) - S$ will induce an r-clique in the resulting graph, $r = |V(H) - S|$).

Given a graph class $\mathscr{G}$, we say that a set $\mathscr{R}$ of reduction rules is *safe* if, for any $R \in \mathscr{R}$ and for any G such that R occurs in G, the result of applying R on G is a graph in $\mathscr{G}$ if and only if $G \in \mathscr{G}$. $\mathscr{R}$ is called *complete* for $\mathscr{G}$, if for every nonempty graph $G \in \mathscr{G}$, there is a reduction rule in $\mathscr{R}$ occurring in G.

Clearly, if a set $\mathscr{R}$ of reduction rules is safe and complete for a graph class $\mathscr{G}$, then, for any graph G, it holds that $G \in \mathscr{R}$ if and only if there exists a sequence of reduction rules in $\mathscr{R}$ that, when successively applied, can reduce G to the empty graph. These reduction rules are in fact a special case of a more general form of reduction rules as studied amongst others in [2] where subgraphs can be rewritten to general graphs, different from a clique.

We denote as $\mathscr{R}_{t \leq 3}$ the set of reduction rules $\{t.i, t.ii, t.iii, t.iv, t.v, t.vi\}$, shown in Fig. 1. For any $R = (H, S, f) \in \mathscr{R}_{t \leq 3}$, S is represented by the white cycles; when $f(v) < \omega$, $f(v)$ is shown.

THEOREM 2 [3, 12, 14]. *$\mathscr{R}_{t \leq 3}$ is a safe and complete set of reduction rules for the class of graphs with treewidth ≤ 3. Also, if we replace rule $t.iv$ in $\mathscr{R}_{t \leq 3}$ by $t.iv'$ the resulting set of rules is also safe and complete for the class of graphs with treewidth ≤ 3.*

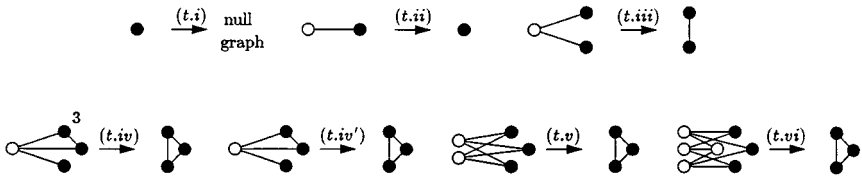

FIG. 1. The reduction rules for the class of graphs with treewidth ≤ 3.

We define below the notions of k-tree, k-perfect elimination ordering, minimal separator, and minimal triangulation.

We call a graph G *chordal* when it does not contain any induced cycle of length at least four. We call a vertex $v \in V(G)$ *simplicial* if $G[N_G(v)]$ is an r-clique of G, where $r = |N_G(v)|$. Let k be a positive integer. An ordering $(v_1, \ldots, v_{|V(G)|})$ of the vertices in $V(G)$ is a *k-perfect elimination ordering* if for each $i, 1 \leq i \leq |V(G)|$ v_i is a simplicial vertex of degree at most k in $G_i = G[\{v_i, \ldots, v_{|V(G)|}\}]$. We call $(G = G_1, G_2, \ldots, G_{|V(G)|})$ the *graph sequence* of the k-perfect elimination ordering.

A k-tree is a graph which is recursively defined as follows. A complete graph with $k + 1$ vertices is a k-tree. Given a k-tree G with $n > k$ vertices, a k-tree with $n + 1$ vertices can be constructed by making a new vertex adjacent to the vertices of a k-clique of G. It is easy to see that k-trees are chordal graphs with maximum clique size $k + 1$. A graph is a partial k-tree if either it has at most k vertices or it is a subgraph of a k-tree G with the same vertex set as G.

It can be easily proved that a graph has treewidth at most k iff it is a partial k-tree (see, e.g., [20]). Also, if G is a partial k-tree, then $|E(G)| < k|V(G)|$. Finally, a k-perfect elimination ordering of a k-tree can be found in $\mathcal{O}(|V(G)|)$ time (k is a fixed constant).

A set $S \subseteq V(G)$ is an *s-t-separator* in G ($s, t \in V(G)$), if s and t belong to different connected components of $G[V - S]$. S is a *minimal s-t-separator*, if it does not contain another s-t-separator as a proper subset. S is a *minimal separator*, if there exist vertices $s, t \in V$ for which S is a minimal s-t-separator. We call a graph G' a *triangulation* of G if G' is chordal and $V(G') = V(G)$. We call a triangulation of G with a minimum number of edges a *minimal triangulation* (see [22]).

THEOREM 3 [6]. *Let G' be a minimal triangulation of a graph G. Then any minimal separator in G' is also a minimal separator in G.*

Consider graphs K_5, M_6, M_8, M_{10}, and Q_3 as shown in Fig. 2.

THEOREM 4 [4, 18]. *The obstruction set for the class of graphs with treewidth at most three, $\mathcal{T}_3$, equals $\{K_5, M_6, M_8, M_{10}\}$.*

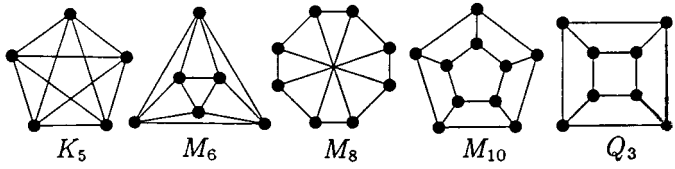

FIG. 2. The graphs K_5, M_6, M_8, M_{10}, and Q_3.

LEMMA 2. *The following three statements hold*:

a. *There are no graphs in* $\mathscr{B}_3$ *with treewidth at most* 2.

b. $Q_3 \in \mathscr{B}_3$ *and* treewidth$(Q_3) = 3$.

c. *The set* $\{K_5, M_6, M_8\}$ *contains all the graphs of* $\mathscr{B}_3$ *that have treewidth at least* 4.

Proof. a. From the first inequality of Theorem 1.b, we have that there are no graphs with treewidth at most 2 and branchwidth at least 4.

b. One can easily verify that treewidth(Q_3) = branchwidth$(Q_3) - 1 = 3$. Also, any graph obtained by Q_3 by a vertex/edge deletion or edge contraction has a branch decomposition of width at most 3.

c. We will first prove that $\{K_5, M_6, M_8\} \subseteq \mathscr{B}_3$. From Theorem 4, $\mathscr{S} = \{K_5, M_6, M_8\} \subseteq \mathscr{T}_3$ and thus $\forall G \in \mathscr{S}$: treewidth$(G) = 4$. From the second inequality of Theorem 1.b, we obtain that $\forall G \in \mathscr{S}$: branchwidth$(G) \geq 4$. It is now enough to check, by inspection, that if we apply to any element of $\mathscr{S}$ a vertex/edge deletion or edge contraction the resulting graph has a branch decomposition of width ≤ 3.

Suppose now that there exists a graph G in $\mathscr{B}_3 - \{K_5, M_6, M_8\}$ that has treewidth ≥ 4. From Theorem 4, G should contain one of the graphs in $\mathscr{T}_3$ as a minor and thus one of the graphs in $\{K_5, M_6, M_8, Q_3\} \subseteq \mathscr{B}_3$ which is a contradiction (observe that $Q_3 \preccurlyeq M_{10}$ and $Q_3 \in \mathscr{B}_3$). ∎

Let G be a graph and $S \subseteq V(G), |S| = 4$. We call $S = \{v_1, v_2, v_3, v_4\}$ a *cross* if each of the sets $S_i = S - \{v_i\}$, $1 \leq i \leq 4$, is a minimal separator of G (for an example, see Fig. 3). If a graph does not contain any cross then we call it *crossless*.

LEMMA 3. *Let* G *be a crossless graph of treewidth at most* 3 *and let* G' *be a minimal triangulation of* G. *Then*, G' *is a crossless chordal graph with maximum clique size at most* 4.

Proof. It is known that if G' is a minimal triangulation of a partial k-tree, then G' has maximum clique size at most $k + 1$ (see, e.g., Chapter 2 of [13]). What remains to prove is that G' is crossless. Suppose that G'

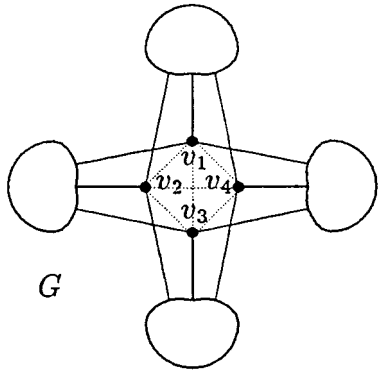

FIG. 3. A graph G containing a cross $S = \{v_1, v_2, v_3, v_4\}$ (notice that S remains a cross even if we remove from G any set of dashed edges).

contain a cross S. Then all the triples of S are minimal separators and, by Theorem 3, they are also minimal separators of G. We have a contradiction as G is crossless. ▮

We now introduce the notion of a clique tree of a 3-tree. (We mention that it is possible to extend the definition below—as well as the algorithm following it—for any integer $k \neq 3$. (For other representations of chordal graphs in tree structures see [11, 23].)

Let G be a 3-tree G. A tree T_G is a *clique tree* of G if

(i) each vertex in $V(T_G)$ is a subset of $V(G)$ inducing a 4-clique in G and

(ii) if two vertices $\mathbf{v} = \{v_1, v_2, v_3, v_4\}$, $\mathbf{u} = \{u_1, u_2, u_3, u_4\} \in V(T_G)$ are connected by an edge $\mathbf{e} = \{\mathbf{v}, \mathbf{u}\}$ in T_G then $|\mathbf{v} \cap \mathbf{u}| = 3$, i.e., they have exactly three vertices in common (notice that each such triple of vertices is a minimal separator of G).

Clearly, a k-tree can have many different clique trees (see Fig. 4). From now on we will denote the vertices and the edges of a clique tree using bold characters like $\mathbf{v}, \mathbf{u}, \mathbf{e}$.

Given an edge $\mathbf{e} = \{\mathbf{v}, \mathbf{u}\} \in E(T_G)$, we define the *separation set* of e as $\mathsf{sep}(\mathbf{e}) = \mathbf{v} \cap \mathbf{u}$. Notice that any clique tree of a 3-tree G contains $|V(G)| - 3$ vertices, one for each 4-clique of G, and $|V(G)| - 4$ edges, one for each minimal separator of G.

We omit the proof of the following as it is easy and follows standard techniques.

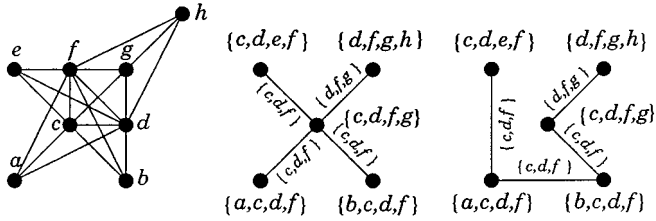

FIG. 4. A 3-tree and two clique trees of it. For any edge **e**, the set sep(**e**) is depicted.

LEMMA 4. *Let* $\mathbf{e}_1, \mathbf{e}_2$ *be two edges of* T_G *and* $v \in V(G)$ *such that* $v \in \mathsf{sep}(\mathbf{e}_1) \cap \mathsf{sep}(\mathbf{e}_2)$. *Then, for any edge* e_3 *of* T_2 *with both endpoints in the path connecting* $\mathbf{e}_1$ *and* $\mathbf{e}_2$ *in* T_G, *we have that* $v \in \mathsf{sep}(e_3)$ *as well*.

We now give an algorithm constructing a clique tree of a 3-tree in linear time.

ALGORITHM. Clique-Tree.
input: A 3-tree G.
output: A clique tree T_G of G.

1. Find a 3-perfect elimination ordering $(v_1, v_2, \ldots, v_n)$ of G ($n = |V(G)|$);
2. **let** $\mathbf{a}_{n-3} = \{v_{n-3}, v_{n-2}, v_{n-1}, v_n\}$;
3. **for** $i = 1$ **to** $n - 4$ **do**
4. **let** $\mathsf{sim}(v_i) = N_{G[V(G)-\{v_1,\ldots,v_{i-1}\}]}(v_i)$;
5. **for** $i = n - 3$ **to** n **do**
6. **let** $\mathsf{sim}(v_i) = \mathbf{a}_{n-3} - \{v_i\}$;
7. **let** $V(T_G) = \{\mathbf{a}_{n-3}\}$, $E(T_G) = \emptyset$;
8. **for** $i := n - 4$ **downto** 1 **do**
9. **begin**
10. **set** $C_i = \bigcup_{v \in \mathsf{sim}(v_i)} \mathsf{sim}(v)$;
11. find a vertex $v_{i'} \in C_i$ such that $\mathbf{a}_{i'} = \mathsf{sim}(v_i) \cup \{v_{i'}\}$ induces a 4-clique in G_i;
12. **let** $\mathbf{a}_i = \mathsf{sim}(v_i) \cup \{v_i\}$;
13. **let** $V(T_G) = V(T_G) \cup \{a_i\}$, $E(T_G) = E(T_G) \cup \{\{\mathbf{a}_{i'}, \mathbf{a}_i\}\}$;
14. **end**
15. **end**

LEMMA 5. *Given a* 3*-tree in* G, *algorithms* Clique-Tree *constructs the clique tree of* G *in* $\mathcal{O}(|V(G)|)$ *time*.

Proof. We will first prove that a vertex as required in Step 11 always exists. Let $\mathsf{sim}(v_i) = \{v_j, v_{j'}, v_{j''}\}$. W.l.o.g. we assume that $j < j', j < j''$. We set $\{v_{i'}\} = \mathsf{sim}(v_j) - \{v_{j'}, v_{j''}\}$ and it is enough to notice that $v_{i'} \in C_j$ and $G[\{v_{i'}, v_j, v_{j'}, v_{j''}\}]$ is a 4-clique of G_i. Moreover, as $\forall_{i \le i \le n} |\mathsf{sim}(v_i)| = 3$, we have that $\forall_{1 \le i \le n-4} |C_i| = \mathcal{O}(1)$ and steps 10 and 11 can be executed in constant time. Therefore, the overall complexity of Clique-Tree is $\mathcal{O}(|V(G)|)$. Observing now how vertices and edges are added in T_G in steps 12 and 13, during each execution of loop 9–14, we can easily see that T_G is a clique tree of G. ∎

The proof of the following lemma is very simple and is omitted from this paper. We just mention that the algorithm involved is based on a traversal of the graph using a 3-perfect elimination ordering of G.

LEMMA 6. *Let G be a chordal graph with maximum clique of size at most 4. One can construct an algorithm that in $\mathcal{O}(|V(G)|)$ time computes all the triconnected components of G.*

LEMMA 7. *Let G be a crossless chordal graph G with maximum clique size 4. Then there exist a crossless 3-tree G' such that G is a subgraph of G' and $V(G') = V(G)$. Moreover, one can construct an algorithm that outputs such a subgraph in $\mathcal{O}(|V(G)|)$ time.*

Proof. We will give a procedure that adds edges to G so that the resulting graph is a crossless 3-tree. As a preprocessing step we apply the following operation until it is not possible any more:

- If G contains an articulation points v then set $G \leftarrow G[V(G), E(G) \cup \{\{v, u\}\}]$, where v, u are two neighbours of v that belong to different connected components of $G[V(G) - \{v\}]$.

Clearly, the above preprocessing can be done in $\mathcal{O}(|V(G)|)$ time and outputs a crossless chordal graph G with maximum clique size at most 4 and with no articulation points.

It is easy to see that, using a 3-perfect elimination ordering, we can compute, in linear time, two functions f_H, g_H such that, for any 3-tree H, f_H takes as input a triangle t of H and outputs a boolean value indicating whether t is a minimal separator of H or not and g_H takes as input an edge e of H and outputs a 4-clique of H containing it. Using the algorithm of Lemma 6, we compute, in linear time, all the triconnected components of G. We also compute for each triconnected component G_i that is a (crossless) 3-tree the corresponding functions f_{G_i} and g_{G_i}.

Suppose now that G_i and G_j are two triconnected components of G. It is enough to show that in constant time, we can (i) add an edge in $G[V(G_i) \cup V(G_j)]$ so that the resulting graph G_{ij} is a crossless 3-tree and

(ii) compute the functions $f_{G_{ij}}$ and $g_{G_{ij}}$ using f_{G_i}, f_{G_j} and g_{G_i}, g_{G_j}. As G does not contain any articulation points we may assume that $V(G_i) \cap V(G_j) = \{a, b\}$. We distinguish the following three cases (see also Fig. 5).

Case (i). G_i and G_j are both crossless 3-trees. Let $g_{G_i}(\{a, b\}) = \{a, b, c, d\}$ and $g_{G_j}(\{a, b\}) = \{a, b, f, e\}$. If $f_{G_i}(\{a, b, c\}) = 0$ then set $v = d$; otherwise set $v = c$. Also, if $f_{G_j}(\{a, b, e\}) = 0$ then set $u = f$; otherwise set $u = e$. Now, construct G_{ij} by adding the edge $\{v, u\}$ in $G[V(G_i) \cup V(G_j)]$. One can now easily verify that G_{ij} is a crossless 3-tree. We set $f_{G_{ij}}(\{a, v, u\}) = f_{G_{ij}}(\{b, v, u\}) = 0$, $f_{G_{ij}}(\{a, b, v\}) = f_{G_{ij}}(\{a, b, u\}) = 1$, and $g_{G_{ij}}(\{v, u\}) = \{a, b, v, u\}$.

Case (ii). G_i is a crossless 3-tree and $V(G_j) = \{a, b, e\}$. Let $g_{G_i}(\{a, b\}) = \{a, b, c, d\}$. If $f_{G_i}(\{a, b, c\}) = 0$ then set $v = d$; otherwise set $v = c$. Now, construct G_{ij} by adding the edge $\{v, e\}$ in $G[V(G_i) \cup V(G_j)]$. One can now easy verify that G_{ij} is a crossless 3-tree. We set $f_{G_{ij}}(\{a, v, e\}) = f_{G_{ij}}(\{b, v, e\}) = f_{G_{ij}}(\{a, b, e\}) = 0$, $f_{G_{ij}}(\{a, b, v\}) = 1$, $g_{G_{ij}}(\{a, e\}) = g_{G_{ij}}(\{b, e\}) = g_{G_{ij}}(\{v, e\}) = \{a, b, v, u\}$.

Case (iii). $V(G_i) = \{a, b, v\}$, $V(G_j) = \{a, b, u\}$. In this case we construct G_{ij} by adding $\{v, u\}$ in $G[V(G_i) \cup V(G_j)]$. Clearly, G_{ij} is a crossless 3-tree. Finally, for any triple t of $\{a, b, v, u\}$ we set $f_{G_{ij}}(t) = 0$ and for any edge $e \subseteq \{a, b, v, u\}$ we set $g_{G_{ij}}(e) = \{a, b, v, u\}$. ▮

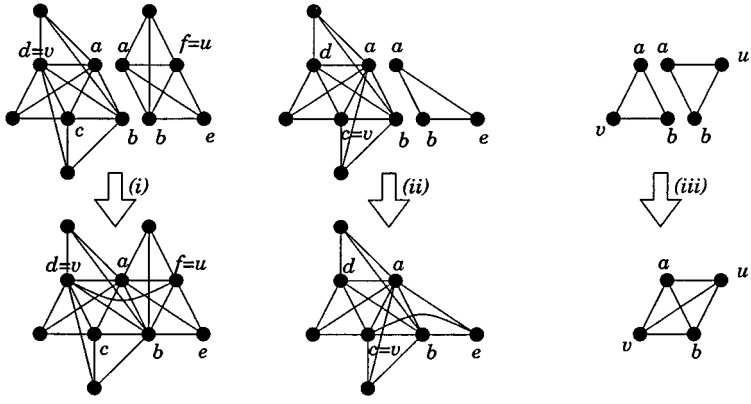

FIG. 5. Examples for cases (i), (ii), and (iii) of the proof of Lemma 7.

3. OBSTRUCTIONS FOR GRAPHS WITH BRANCHWIDTH AT MOST 3

In this section we will identify the obstruction set for the class of graphs with branchwidth at most three. Unless stated otherwise, we consider a clique tree T_G of a crossless 3-tree G.

Given a k-tree, a clique tree T_G of G, and a vertex $\mathbf{v} \in V(T_G)$, we define $E_{\mathbf{v}} = \{\mathbf{e} \in E(T_G)$: $\mathbf{v}$ belongs to $\mathbf{e}\}$. The following lemma defines the notion of the 3-labelled clique tree of a crossless 3-tree.

LEMMA 8. *Let G be a crossless 3-tree G and T_G a clique tree of G. Then, there exist a function l: $E(T_G) \to \{1, 2, 3\}$ such that $\forall \mathbf{v} \in V(T_G)$: $\forall \mathbf{e}_1, \mathbf{e}_2 \in E_{\mathbf{v}}$: $(\mathsf{sep}(\mathbf{e}_1) = \mathsf{sep}(\mathbf{e}_2)$ iff $l(\mathbf{e}_1) = l(\mathbf{e}_2))$; i.e., edges in $E_{\mathbf{v}}$ with the same separation set have the same label. Moreover, one can construct an algorithm that, given T_G, computes such a function l in $\mathcal{O}(|V(G)|)$ time.*

Proof. Since G is crossless, we have that, for any 4-clique of G, at most three of its triples are minimal separators and therefore $\forall \mathbf{v} \in V(T_G)$: $|\{\mathsf{sep}(\mathbf{e}) | \mathbf{e} \in E_{\mathbf{v}}\}| \leq 3$. It is now possible, for any vertex $\mathbf{v}$ of T_G, to compute in $\mathcal{O}(|N_{T_G}(\mathbf{v})|)$ time, a partition of $E_{\mathbf{v}}$ into the minimum number of sets containing edges with the same separation set (we call such a partition *a separating partition* of $\mathbf{v}$). We can now label the edges of T_G as follows. We first label arbitrarily an edge containing a leaf of T_G. Suppose now that we have labelled all the edges containing vertices in some set $V' \subset V(T_G)$. As T_G is connected, there must exists at least one vertex $\mathbf{v} \in V(T_G) - V'$ such that one of the edges containing it has already been labelled. Now, using the separating partition of $\mathbf{v}$, we can label its edges such that edges belonging to the same set of the partition have the same label. It is now clear that such a labelling can be computed in $\mathcal{O}(|V(T_G)|)$ time. ∎

We call a clique tree of a chordless 3-tree that is labelled as in Lemma 8, 3-*labelled*, and we denote it as (T_G, l). Given a 3-labelled clique tree (T_G, l), we define the *span-degree* of a vertex $\mathbf{v}$ to be equal to $|\{l(\mathbf{e})$: $\mathbf{e} \in E_{\mathbf{v}}\}|$. As an example notice that in both 3-labelled clique trees of Fig. 6 all the vertices have span-degree 1 except from $\{c, d, f, g\}$ that has span-degree 3.

LEMMA 9. *Any crossless 3-tree G has an amplified branchwidth decomposition of width 3. Moreover, one can construct an algorithm that, given a 3-labelled tree clique (T_G, l) of G, outputs an amplified branch decomposition of G with width 3 in $\mathcal{O}(|V(G)|)$ time.*

Proof. We examine the non trivial case where $|E(T_G)| > 0$. We will give a construction that, given a 3-labelled clique tree (T_G, l) of a crossless

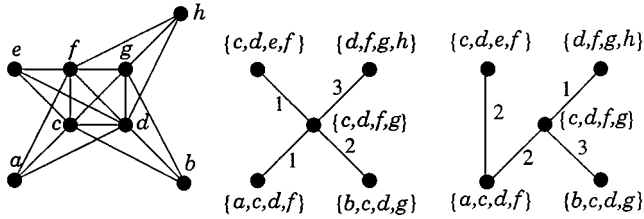

FIG. 6. A 3-tree and two 3-labelled clique trees of it.

3-tree G, outputs an amplified branch decomposition (T, τ) of G that has width 3. We will first construct T. Let $\mathbf{v}$ be a vertex of T_G with span-degree d. Using function l we can assume that $E_{\mathbf{v}} = E^1_{\mathbf{v}} \cup \cdots \cup E^d_{\mathbf{v}}$, where $E^i_{\mathbf{v}} = \{\mathbf{e}^i_1, \ldots, \mathbf{e}^i_{r_i}\}$, $i = 1, \ldots, d$, and $\forall i, 1 \le i \le d$: $\forall j, 1 \le j \le r_i$: $l(e^i_j) = i$; i.e., we partition $E_{\mathbf{v}}$ into a minimum number of sets each containing edges with the same label. Notice that to find such a partition for all the vertices of T_G costs $\mathcal{O}(|V(T_G)|)$ time.

For any vertex $\mathbf{v} \in V(T_G)$, we construct the tree $T_{\mathbf{v}} = (V(T^1_{\mathbf{v}}) \cup V(T^2_{\mathbf{v}}) \cup V(T^3_{\mathbf{v}})), E(T^1_{\mathbf{v}}) \cup E(T^2_{\mathbf{v}}) \cup E(T^3_{\mathbf{v}}))$ where, for $i = 1, 2, 3$;

- In case $i \le d$ and $r_i \ge 2$, then we set

$$T^i_{\mathbf{v}} = \Bigg(\{v_0\} \cup \{v^i_1, \ldots, v^i_{r_i - 1}\} \bigcup_{1 \le j \le r_i} \{V(T^{i,j}_{\mathbf{v}})\},$$

$$\{\{v_0, v^i_1\}, \{v^i_1, v^i_2\}, \ldots, \{v^i_{r_i-2}, v^i_{r_i-1}\}\} \bigcup_{1 \le j \le r_i} \{E(T^{i,j}_{\mathbf{v}})\} \Bigg),$$

where for $j = 1, \ldots, r_i - 1$,

$$T^{i,j}_{\mathbf{v}} = \Big(\{v^i_j, v^i_{j,1}, v^i_{j,2}, v^i_{j,3}, v^i_{j,4}, v^i_{j,5}, v^i_{j,6}\},$$

$$\{\{v^i_j, v^i_{j,1}\}, \{v^i_{j,1}, v^i_{j,2}\}, \{v^i_{j,2}, v^i_{j,3}\}, \{v^i_{j,1}, v^i_{j,4}\},$$

$$\{v^i_{j,2}, v^i_{j,5}\}, \{v^i_{j,3}, v^i_{j,6}\}\}\Big) \text{ and}$$

$$T^{i,r_i}_{\mathbf{v}} = \Big(\{v^i_{r_i-1}, v^i_{r_i,1}, v^i_{r_i,2}, v^i_{r_i,3}, v^i_{r_i,4}, v^i_{r_i,5}, v^i_{r_i,6}\},$$

$$\{\{v^i_{r_i-1}, v^i_{r_i,1}\}, \{v^i_{r_i,1}, v^i_{r_i,2}\}, \{v^i_{r_i,2}, v^i_{r_i,3}\}, \{v^i_{r_i,1}, v^i_{r_i,4}\},$$

$$\{v^i_{r_i,2}, v^i_{r_i,5}\}, \{v^i_{r_i,3}, v^i_{r_i,6}\}\}\Big).$$

The tree $T_{\mathbf{v}}$ is illustrated in Fig. 7. We call the path $(v_0, v_1^i, \ldots, v_{r_i-1}^i)$ *the trunk* and the vertices in $\{v_1^i, \ldots, v_{r_i-1}^i\}$ *the body* of the trunk.

- In case $i \leq d$ and $r_i = 1$ we define T_v^i by applying the construction given above for $r_i = 1$ with the difference that, where in that construction $v_{r_i-1}^i$ is used, now v_0 is used. (Clearly, in such a case, the trunk of $T_{\mathbf{v}}^i$ contains only v_0.)
- In case $i > d$, we set

$$T_{\mathbf{v}}^i = \left(\left\{v_0, v_1^i, v_2^i, v_3^i, v_4^i, v_5^i\right\}, \left\{\left\{v_0, v_1^i\right\}, \left\{v_1^i, v_2^i\right\}, \left\{v_2^i, v_3^i\right\}, \left\{v_1^i, v_4^i\right\}, \left\{v_2^i, v_5^i\right\}\right\}\right).$$

If $i > d$, we call the subtree $T_{\mathbf{v}}^i$ constructed above *external*; otherwise, if $i \leq d$, we call the subtrees $T_{\mathbf{v}}^{i,1}, \ldots, T_{\mathbf{v}}^{i,r_i}$ *internal subtrees* of $T_{\mathbf{v}}$. We also call vertex v_0 *the kernel*. Observe now that each of the internal subtrees of $T_{\mathbf{v}}$ that comprise $T_{\mathbf{v}}^i$ corresponds to one of the edges in $E_{\mathbf{v}}$ with label i; i.e., edge $\mathbf{e}_j^i$ corresponds to the tree $T_{\mathbf{v}}^{i,j}$ for $1 \leq j \leq r_i, 1 \leq i \leq d$.

For an example illustrating the three cases above see Fig. 8. In the clique tree of Fig. 8 vertex $\mathbf{v}$ has span-degree 2 and belongs to edges $e_1^1, e_2^1, e_3^1, e_4^1, e_1^2$, where $l(e_j^1) = 1$, $1 \leq j \leq 4$, and $l(e_1^2) = 2$. Clearly, $T_{\mathbf{v}}^1$ consists of a trunk $\{v_0, v_1^1, v_2^1, v_3^1\}$ and four internal subtrees, $T_{\mathbf{v}}^{1,1}$, $T_{\mathbf{v}}^{1,2}$, $T_{\mathbf{v}}^{1,3}$, and $T_{\mathbf{v}}^{1,4}$ of $T_{\mathbf{v}}$, each corresponding to one of the edges of $E_{\mathbf{v}}$ labelled with 1. As the unique edge labelled with 2 is e_1^2, the subtree corresponding to it is $T_{\mathbf{v}}^2 = T_{\mathbf{v}}^{2,1}$ which is an internal subtree of $T_{\mathbf{v}}$. Finally, the fact that there are no edges labelled with 3 implies the existence of subtree $T_{\mathbf{v}}^3$ in $T_{\mathbf{v}}$. Clearly, $T_{\mathbf{v}}^3$ is an external subtree.

The construction of the tree T of the amplified branch decomposition is now completed by applying, for any edge $\mathbf{e}$ of T_G, the following operation: Suppose that $\mathbf{e} = \{\mathbf{v}, \mathbf{u}\}$. Let also $T_{\mathbf{v}}$ and $T_{\mathbf{u}}$ be the subtrees corresponding to $\mathbf{v}$ and $\mathbf{u}$, according to the construction described above. We denote as $T_{\mathbf{v}}^{i_{\mathbf{v}}, j_{\mathbf{v}}}$ the internal subtree of $T_{\mathbf{v}}$ corresponding to edge $\mathbf{e}$ and as $T_{\mathbf{u}}^{i_{\mathbf{u}}, j_{\mathbf{u}}}$ the internal subtree of $T_{\mathbf{u}}$ corresponding to edge e. The construction proceeds by identifying the pairs of vertices $(v_{j_{\mathbf{v}},1}^{i_{\mathbf{v}}}, u_{j_{\mathbf{u}},3}^{i_{\mathbf{u}}})$, $(v_{j_{\mathbf{v}},2}^{i_{\mathbf{v}}}, u_{j_{\mathbf{u}},2}^{i_{\mathbf{u}}})$, $(v_{j_{\mathbf{v}},3}^{i_{\mathbf{v}}}, u_{j_{\mathbf{u}},1}^{i_{\mathbf{u}}})$, $(v_{j_{\mathbf{v}},4}^{i_{\mathbf{v}}}, u_{j_{\mathbf{u}},6}^{i_{\mathbf{u}}})$, $(v_{j_{\mathbf{v}},5}^{i_{\mathbf{v}}}, u_{j_{\mathbf{u}},5}^{i_{\mathbf{u}}})$, $(v_{j_{\mathbf{v}},6}^{i_{\mathbf{v}}}, u_{j_{\mathbf{u}},4}^{i_{\mathbf{u}}})$ and eliminating the double edges that

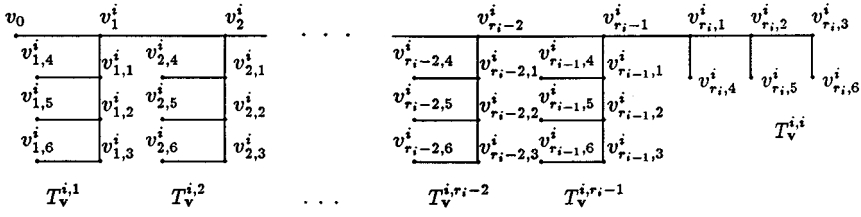

FIG. 7. The tree $T_{\mathbf{v}}^i$.

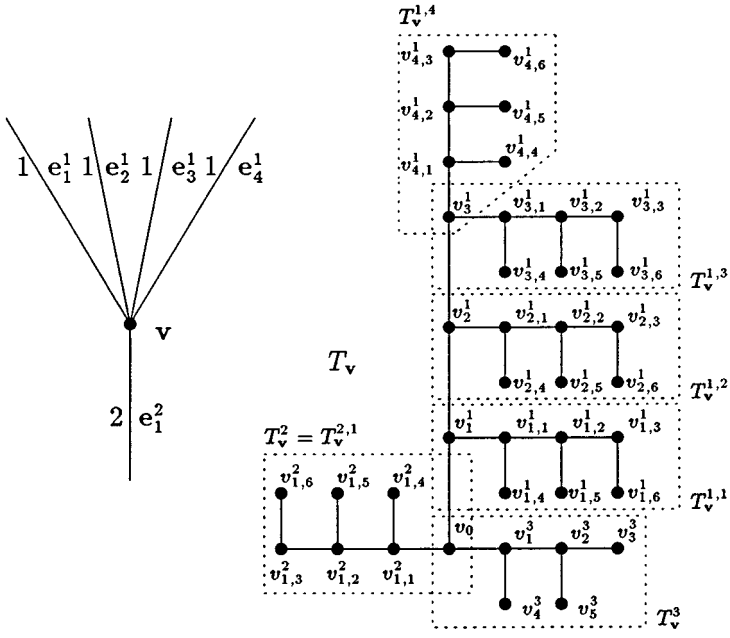

FIG. 8. A vertex **v** in a clique tree, and the corresponding tree T_v.

appear (see also Fig. 9.) We call the above operation *the merging operation* and notice that it is applied always to internal subtrees.

Notice now that, applying the merging operation for any edge of T_G, we construct a tree with vertices of degree 1 or 3. Moreover, such a construction can be done in $\mathcal{O}(V(T_G))$ time. Notice that each vertex $\mathbf{v} \in V(T_G)$

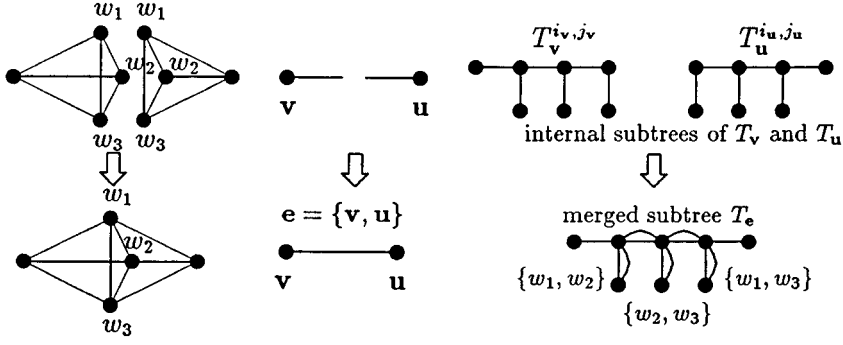

FIG. 9. Identifying vertices of $T_{\mathbf{v}}^{i_{\mathbf{v}},j_{\mathbf{v}}}$ and $T_{\mathbf{u}}^{i_{\mathbf{u}},j_{\mathbf{u}}}$ towards constructing $T_{\mathbf{e}}$.

corresponds to exactly one kernel, say $v_0^{\mathbf{v}}$, in $V(T)$. Moreover, each edge $\mathbf{e} = \{\mathbf{v}, \mathbf{u}\} \in E(T_G)$, corresponds to exactly one subtree of T that is the result of merging two internal subtrees $T_{\mathbf{v}}^{i_{\mathbf{v}}, j_{\mathbf{v}}}, T_{\mathbf{u}}^{i_{\mathbf{u}}, j_{\mathbf{u}}}$ of $T_{\mathbf{v}}$ and $T_{\mathbf{u}}$, respectively. We call the product of this merging *the merged subtree* of T and we denote it as $T_{\mathbf{e}}$. We also call the nonleaf vertices of a merged subtree *central* vertices. Given a trunk L of T we call *an L-merged subtree* any merged subtree sharing a common vertex with the body of L. We say that two merged subtrees $T_{\mathbf{e}_i}$, $i = 1, 2$, of T are *touching* if the path connecting their vertex sets contains only vertices of trunks of T. According to the construction above, two edges $\mathbf{e}_1, \mathbf{e}_2$ of $E(T_G)$ have a common vertex in T_G iff $T_{\mathbf{e}_1}$ and $T_{\mathbf{e}_2}$ are touching. A direct consequence of that is the following fact.

Fact A. Let $\mathbf{e}_i \in E(T_G)$, $i = 1, 2, 3$. Then, both endpoints of e_3 belong to the path that connects $\mathbf{e}_1$ and $\mathbf{e}_2$ in T_G iff the central vertices of $T_{\mathbf{e}_3}$ belong to the path that connect the vertex sets of $T_{\mathbf{e}_1}$ and $T_{\mathbf{e}_2}$.

What now remains is to define the function τ: leaves of $T \to E(G)$. We distinguish two kinds of leaves in T: those that are leaves of merged subtrees and those that belong to external subtrees. In the first case we have to define τ for leaves appearing in triples of the form $v_{j,4}^i, v_{j,5}^i, v_{j,6}^i$. Recall that each such triple corresponds to some edge $\mathbf{e}$ of the clique tree of G. Let $\mathsf{sep}(\mathbf{e}) = \{w_1, w_2, w_3\}$. We define $\tau(v_{j,4}^i) = \{w_1, w_2\}$, $\tau(v_{j,5}^i) = \{w_2, w_3\}$, and $\tau(v_{j,6}^i) = \{w_1, w_3\}$ (see Fig. 9). Now, we can observe the following fact.

Fact B. If p_1, p_2, p_3 are the leaves of a merged subtree $T_{\mathbf{e}}$ of T then $\tau(p_i) \subset \mathsf{sep}(\mathbf{e})$, $i = 1, \ldots, 3$.

We claim that, for any edge e that is a subset of a minimal separator S of G, there is a leaf v in T where $\tau(v) = e$. Indeed, v should be one of the leaves of $T_{\mathbf{e}}$ where $\mathbf{e}$ is the edge of T_G for which $\mathsf{sep}(\mathbf{e}) = S$.

The only leaves of T, where τ is still undefined, are the leaves of the external subtrees of T. Let $T_{\mathbf{v}}^i$ be such an external subtree and let v_3^i, v_4^i, v_5^i be its leaves. Let d be the span-degree of $\mathbf{v}$. We distinguish the cases (see also Fig. 10):

Case (i). $d = 2$: $\{w_1, w_2, w_3, w_4\}$ contains exactly two triples, say $t_1 = \{w_1, w_2, w_4\}$, $t_2 = \{w_2, w_3, w_4\}$ that are minimal separators of G. Clearly, $i = d + 1 = 3$ and we set $\tau(v_3^3) = \{w_1, w_3\}$, $\tau(v_4^3) = \{w_1, w_3\}$, and $\tau(v_5^3) = \{w_2, w_3\}$.

Case (ii). $d = 1$: $\{w_1, w_2, w_3, w_4\}$ contains only one triple, say $t_1 = \{w_2, w_3, w_4\}$, that is a minimal separator of G. Clearly, $i = 2$ or 3. W.l.o.g. we assume that $i = 2$ and observe that $T_{\mathbf{v}}$ contains another external subtree $T_{\mathbf{v}}^3$. Let $\{v_3^3, v_4^3, v_5^3\}$ be the leaves of $T_{\mathbf{v}}^3$. We set $\tau(v_3^2) = \{w_1, w_3\}$,

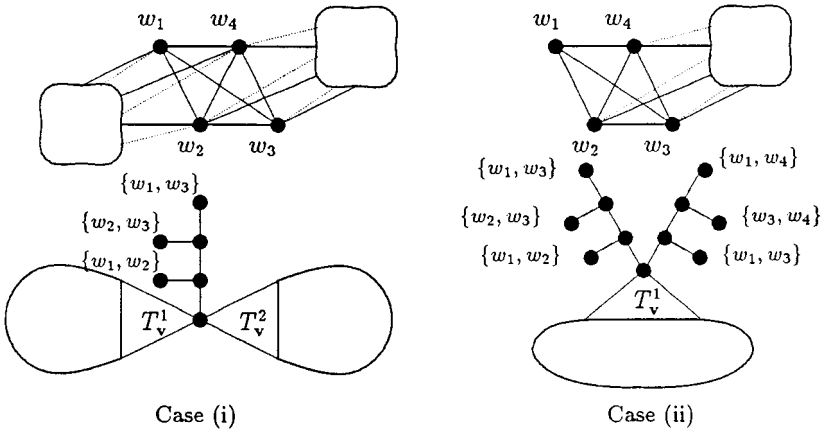

FIG. 10. An illustration of the two cases in the proof of Lemma 9.

$\tau(v_4^2) = \{w_1, w_2\}$, $\tau(v_5^2) = \{w_2, w_3\}$, $\tau(v_3^3) = \{w_1, w_4\}$, $\tau(v_4^3) = \{w_1, w_3\}$, and $\tau(v_5^3) = \{w_3, w_4\}$.

We now claim that if an edge e of G does not belong to any minimal separator there will be an external subtree $T_{\mathbf{v}}^i$ containing a leaf v, where $\tau(v) = e$. Using Lemma 4 one can see that e will be the subset of exactly one vertex $\mathbf{v}$ of T_G. Let $v_0^{\mathbf{v}}$ be the kernel of T corresponding to $\mathbf{v}$. Notice that $v_0^{\mathbf{v}}$ is a vertex of at most two external subtrees of T and, according to the definition of τ in Cases (i) and (ii) above, v will be one of their leaves.

We now have that any edge in G is an image of some leaf of T and thus (T, τ) is an amplified branch decomposition of G. Moreover, the construction of G can be done in $\mathcal{O}(|V(G)|)$ time.

What now remains is to prove that the width of (T, τ) is 3. Let e be an edge of T. If e belongs to an external subtree of T, it is trivial that it has order ≤ 3. Therefore, we can assume that either e is an edge of a merged subtree of T or that it belongs to a trunk of T. Let p_i, $i = 1, 2$, be two leaves of T each belonging to a different connected component of $(V(T), E(T) - \{e\})$. Let also $v \in \tau(p_1) \cap \tau(p_2)$.

If e belongs to a merged subtree of T, we denote it as $T_{\mathbf{e}}$, where $\mathbf{e}$ is the corresponding edge of T_G. We define as P_i the path connecting e with p_i in T for $i = 1, 2$. In case e belongs to a trunk L of T, we define $T_{\mathbf{e}}$ as the L-merged subtree of T that has at least one of its central vertices, either in P_1 or in P_2 (it is easy to check that such an L-merged subtree always exists). Clearly, as $|\mathsf{sep}(\mathbf{e})| = 3$, it is sufficient to prove that $v \in \mathsf{sep}(\mathbf{e})$.

We may assume that neither p_1 nor p_2 is a leaf of $T_{\mathbf{e}}$ (otherwise, the required statement follows from Fact B). Notice that p_1, p_2 belong to different subtrees T_1, T_2 (merged or external) of T.

We examine first the nontrivial case, where both T_1, T_2 are merged subtrees, and we denote them as $T_{\mathbf{e}_1}$ and $T_{\mathbf{e}_2}$, respectively. Notice that all the central vertices of $T_{\mathbf{e}}$ belong to P_1 or P_2 and, thus, in the path of T that connects p_1 with p_2. Applying Fact A, we have that both endpoints of $\mathbf{e}$ belong to the path connecting $\mathbf{e}_1$ and $\mathbf{e}_2$ in T_G. From Lemma 4 we have that $\mathsf{sep}(\mathbf{e}_1) \cap \mathsf{sep}(\mathbf{e}_2) \subseteq \mathsf{sep}(\mathbf{e})$. Using now Fact B, we have that $v \in \tau(p_i) \subset \mathsf{sep}(\mathbf{e}_i)$, $i = 1, 2$, and thus $v \in \mathsf{sep}(\mathbf{e})$.

We now assume that one, say T_1, of T_1, T_2 is an external tree. Notice that, according to the definition of τ for the leaves of the external subtrees, at least one, say T_1', of the merged subtrees that touch T_i should have a leaf p_1' such that $v \in \tau(p_1')$. If p_1' is a leaf of $T_{\mathbf{e}}$ then the required follows from Fact B; otherwise, it is enough to apply the nontrivial case for p_1' and p_2. The case where both T_1, T_2 are external trees is very similar. ∎

For an example of the construction presented in the proof of Lemma 9, see Figs. 11(i)–(iii). We have now the following.

THEOREM 5. *The following propositions are equivalent*:

a. *A graph G has branchwidth at most* 3.

b. *G has treewidth at most* 3 *and $Q_3 \not\leq G$.*

c. *G has treewidth at most* 3 *and G is crossless.*

Proof. (a ⇒ b) Suppose that branchwidth$(G) \leq 3$. Then, from the second inequality of Theorem 1.b, we get that treewidth$(G) \leq 3$. We also have the $Q_3 \not\leq G$ because, otherwise from Theorem 1.a we have that branchwidth$(G) \geq$ branchwidth$(Q_3) = 4$, a contradiction.

(b ⇒ c) It is sufficient to prove that if a graph G has a cross, then it contains Q_3 as a minor. Let $S = \{v_1, v_2, v_3, v_4\}$ be a cross in G. It is now enough to see that if we contract all edges of G, except those that have at least one endpoint in S, we obtain Q_3.

(c ⇒ a) From Lemmata 3 and 7, we have that G is a subgraph of a crossless 3-tree G. Applying now Lemmata 9 and 1, we have the required. ∎

THEOREM 6. *The obstruction set for the class of graphs with branchwidth at most three, $\mathscr{B}_3$, equals* $\{K_5, M_6, M_8, Q_3\}$.

Proof. By Lemma 2, it is sufficient to prove that Q_3 is the only element of $\mathscr{B}_3$ with treewidth equal to 3. That statement holds because, according to Theorem 5, any graph treewidth at most 3 that does not contain Q_3 as a minor, has branchwidth at most 3. ∎

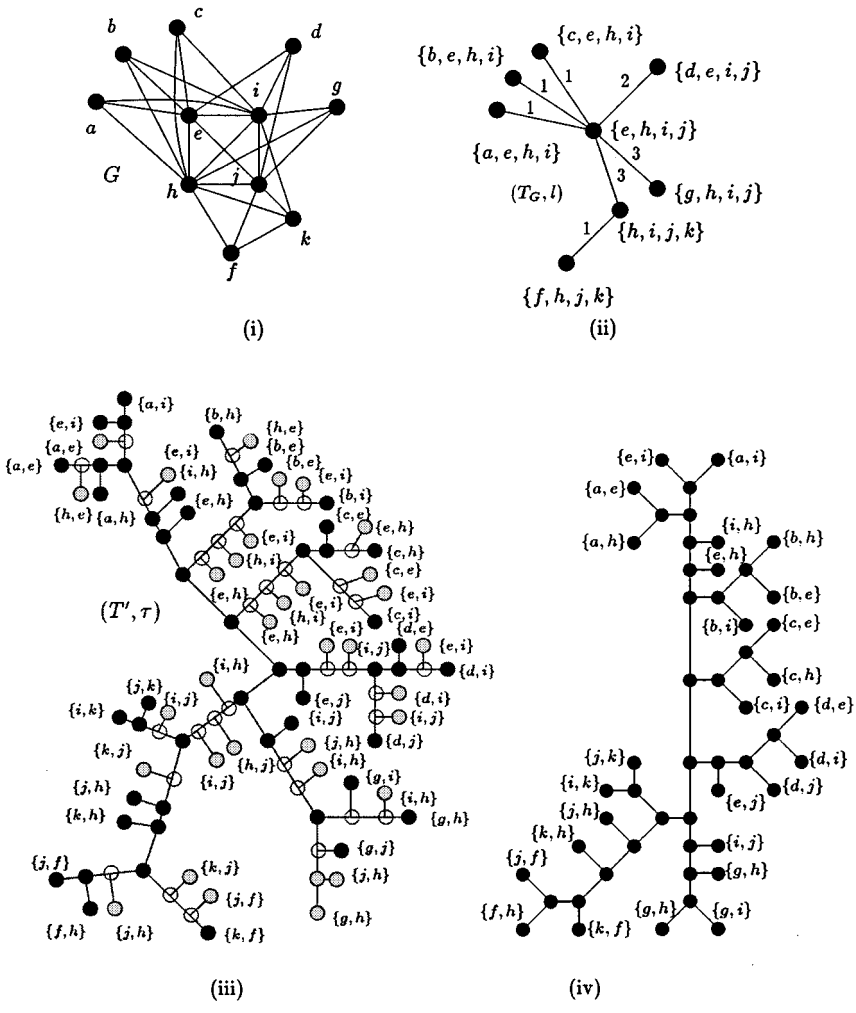

FIG. 11. (i) A crossless 3-tree G, (ii) its 3-labelled clique tree (T_G, l), (iii) an amplified branch decomposition (T, τ) as it is constructed by the algorithm of Lemma 9, and (iv) the branch decomposition constructed by (T, τ), using the algorithm of Lemma 1.

4. REDUCTION RULES FOR GRAPHS WITH BRANCHWIDTH AT MOST 3

In this section we give a safe and complete set of reduction rules for the graphs with branchwidth at most three. (The same conventions are used as in Fig. 1.) We denote as $\mathscr{R}_{b \leq 3}$ the set of reduction rules shown in Fig. 12.

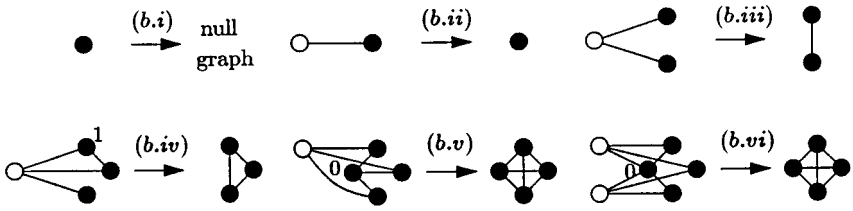

FIG. 12. The reduction rules for the class of graphs with branchwidth ≤ 3.

LEMMA 10. *Let G be a graph with branchwidth at most* 3. *Suppose also that rule b.v or rule b.vi occurs in G. Then, if we apply b.v or b.vi on G the resulting graph has also branchwidth at most* 3.

Proof. We will examine together the two cases where we apply $b.v$ or $b.vi$. Suppose that for some graph G, where branchwidth$(G) \leq 3$, the application of rule R, that is either $b.v$ or $b.vi$, on G results in a graph G' with branchwidth$(G') > 3$. From the second inequality of Theorem 1.b we have that treewidth$(G) \leq 3$. Let $\{a, b, c, d\}$ be the vertices of the resulting 4-clique of G'. W.l.o.g. we can assume that $N_{G'}(a) = \{b, c, d\}$. Observe that $H = G'[V(G') - \{a\}]$ has treewidth at most 3, since treewidth$(G) \leq 3$ and H is the result of a single application of rule $t.v$, in case R is $b.v$, or of three successive applications of rule $t.iv'$, in case R is $b.vi$, on G. Moreover, G' has also treewidth at most 3, as $\{b, c, d\}$ induces a 3-clique in H (e.g., see [8] or [13]). Now, from Theorem 5, we have that G' contains a cross S. By the definition of the cross we have that for any vertex $v \in S$ there must exist three vertices in $N_{G'}(v)$ forming an independent set of G'. Therefore $a \notin S$. We also claim that $|\{b, c, d\} \cap S| \leq 1$. Suppose to the contrary that w.l.o.g. $\{b, c\} \subseteq S$. Then, S would be a cross also in $G'' = G'[V(G'), E(G') - \{\{b, c\}\}]$ and this is a contradiction as $G'' \preccurlyeq G$. We can now assume w.l.o.g. that $\{b, c\} \cap S = \emptyset$. Therefore, $\{a, b, c\}$ belongs to the vertex set of one of the connected components of $G'[V(G') - S]$. The same connected component remains connected even if we remove $\{b, c\}$ from G' as a is adjacent to both b, c. Therefore, graph $G'' = G'[V(G'), E(G') - \{\{b, c\}\}]$ contains a cross which is a contradiction as $G'' \preccurlyeq G$. ∎

LEMMA 11. *$\mathscr{R}_{b \leq 3}$ is a safe set of reduction rules for the class of graphs with branchwidth bounded by* 3.

Proof. We have to prove that the application of any reduction rule from $\mathscr{R}_{b \leq 3}$ to a graph preserves both its membership and nonmembership. Membership for $b.v$ and $b.vi$ holds immediately from Lemma 10. For the rest of the rules in $\mathscr{R}_{b \leq 3}$ membership is easy because of Theorem 1.a,

as the application of any of them on a graph G with branchwidth ≤ 3 results a graph H, where $H \preccurlyeq G$.

Suppose now to the contrary that there exists a graph G with branchwidth ≥ 4 and a reduction rule $R \in \mathscr{R}_{b \leq 3}$ occurring in G, such that if G' is the result of applying R on G, then G' has branchwidth ≤ 3. Notice that, according to Theorem 5, G' is crossless and has treewidth ≤ 3. We also have from the same theorem that either G has treewidth ≥ 4 or it contains a cross. Suppose that G has treewidth ≥ 4. Notice that, as $\mathscr{R}_{t \leq 3}$ is safe, there are no rules in $\mathscr{R}_{t \leq 3}$ occurring in G. Clearly R cannot be *b.i* (or *b.ii*, or *b.iii*, or *b.iv*); otherwise rule *t.i* (*t.ii*, or *t.iii*., or *t.iv*) would occur in G). Moreover, it is not hard to see that if R is *b.v* (*b.vi*), then also *t.v* (*t.iv*) occurs in G, a contradiction. So, G has treewidth at most 3 and, therefore, contains a cross. Recall that G' has also treewidth at most 3 and is crossless. Let $S = \{a, b, c, d\}$ be a cross in G but not in G'. Notice that G' cannot result after the application of rules *b.i*, *b.ii*, *b.iii*, or *b.v* on G, as those applications cannot harm the status of S as a cross. Thus, R is either *b.iv* or *b.vi*. In the first case G contains one vertex more than G' which is adjacent to the vertices of a triple, say $\{b, c, d\}$ of S. This case leads to a contradiction because b, c, and d are all adjacent to more than one vertex in $V(G'[V(G') - \{b, d, c\}])$ and rule *b.iv* cannot be applied. In the second case G should contain two vertices more than G', each one adjacent to different triples in $\{a, b, c, d\}$. In this case we have again a contradiction because a, b, c, d are all adjacent to more than zero vertices in $G'[V(G') - \{a, b, c, d\}]$ and rule *b.vi* cannot be applied. ▌

We call a leaf $\mathbf{u}$ of T_G that is adjacent to a vertex $\mathbf{v}$ *simple* if all the edges of T_G connecting $\mathbf{v}$ with leaves of T_G have pairwise different labels. For example, notice that in the first 3-labelled clique tree of Fig. 6 leaves $\{d, f, g, h\}$ and $\{b, c, d, g\}$ are simple. Moreover, in the second 3-labelled clique tree of Fig. 6 all the leaves are simple.

LEMMA 12. *Let (T_G, l) be a 3-labelled clique tree containing at least one edge. Then, one of the following holds*:

(i) *There exists at least one nonsimple leaf.*

(ii) *There exists a simple leaf* $\mathbf{u}$ *in* T_G *adjacent to a vertex* $\mathbf{v}$ *of span-degree* 1.

(iii) *There exists a simple leaf* $\mathbf{u}$ *in* T_G *adjacent to a vertex* $\mathbf{v}$ *of span-degree* 2.

(iv) *There exist two simple leaves* $\mathbf{u}_1$ *and* $\mathbf{u}_2$ *in* T_G *adjacent to a vertex* $\mathbf{v}$ *of span-degree* 3.

Proof. Let L_0 be the set of leaves of T_G. Let also $L_1 = \bigcup_{\mathbf{v} \in L_0} N_{T_G}(\mathbf{v})$. Suppose now that $L_1 \neq \emptyset$; otherwise, (ii) trivially holds. Moreover, we

assume that all the leaves in T_G are simple; otherwise (i) holds. As $T_G[V(T_G) - L_0]$ is a nonempty tree, L_1 must contain at least one leaf $\mathbf{v}$. Clearly, $\mathbf{v}$ is adjacent to at most d, $2 \leq d \leq 4$, vertices in T_G and only one of them is not a leaf. It is now easy to verify that if the span-degree of $\mathbf{v}$ is 1, 2, or 3; then (ii), (iii), (iv) respectively holds (see also Fig. 13). ▮

LEMMA 13. *Let G be a crossless* 3*-tree. Then, there exists one reduction rule in $\mathscr{R}_{b \leq 3}$ occurring in G.*

Proof. Let (T_G, l) be a 3-labelled clique tree of G. As the lemma is obvious for the case where G is isomorphic to K_4, we may assume that $E(T_G) > 0$. Using Lemma 12, we distinguish the cases (see also Fig. 13):

Case i. T_G contains a leaf $\mathbf{u}_1$ adjacent to a vertex $\mathbf{v}$ that is also adjacent to a vertex $\mathbf{u}_2$ and such that $l(\{\mathbf{v}, \mathbf{u}_1\}) = l(\{\mathbf{v}, \mathbf{u}_2\})$. In this case we can easily see that the two vertices in the set $\mathbf{u}_1 \cup \mathbf{u}_2 - \mathsf{sep}(\{\mathbf{v}, \mathbf{u}_1\})$ are adjacent only to the three vertices in $\mathsf{sep}(\{\mathbf{v}, \mathbf{u}_1\}) = \mathsf{sep}(\{\mathbf{v}, \mathbf{u}_2\}$ and, thus, rule $b.v$ can be applied.

Case ii. T_G contains a simple leaf $\mathbf{u}$ adjacent to a vertex $\mathbf{v}$ with span-degree 1. The case where $\mathbf{u}$ is the only vertex of T_G adjacent to $\mathbf{v}$ is trivial as both rules $b.iv, b.v$ can be applied. If now there exists a vertex $\mathbf{u}' \neq \mathbf{u}$ adjacent to $\mathbf{v}$ and $\{\mathbf{v}, \mathbf{u}\}$ and $\{\mathbf{v}, \mathbf{u}'\}$ have the same label, it is clear that the only triple of $\mathbf{v}$ that is a minimal separator of G is $S = \mathsf{sep}(\{\mathbf{v}, \mathbf{u}\})$. Notice now that the unique vertices in $\mathbf{v} - S$ and $\mathbf{u} - S$ are adjacent only to the vertices of S in G and, therefore, rule $b.v$ can be applied.

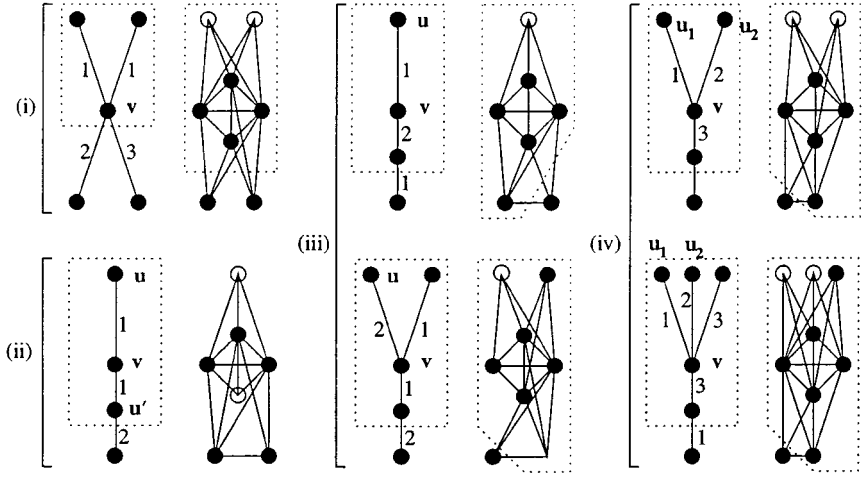

FIG. 13. Examples of the cases of Lemmata 12 and 13.

Case iii. T_G contains a simple leaf $\mathbf{u}$ adjacent to a vertex $\mathbf{v}$ with span-degree 2. We observe that $\mathbf{v}$ contains exactly two triples S_1, S_2 that are minimal separators and one of them, say S_1, contains vertices that are all adjacent to the single vertex in $\mathbf{u} - \mathbf{v}$. Observe that, as $\mathbf{u}$ is simple, the single vertex in $\mathbf{v} - S_2$ is adjacent to exactly one vertex not in $\mathbf{u}$ and thus rule *b.iv* can be applied.

Case iv. There exist two simple leaves $\mathbf{u}_1, \mathbf{u}_2$ in T connected with the same vertex $\mathbf{v}$ that has span-degree 3. In this case, $\mathbf{v}$ contains exactly 3 triples S_1, S_2, S_3 that are minimal separators and two of them, denote them S_1, S_2, contain vertices that are all adjacent to the single vertex in $\mathbf{u}_1 - \mathbf{v}$ and to the single vertex $\mathbf{u}_2 - \mathbf{v}$, respectively. It is easy to see that, as $\mathbf{u}$ is simple, the single vertex in $\mathbf{v} - S_3$ is adjacent only to the vertices in $\mathbf{v} \cup \mathbf{u}_1 \cup \mathbf{u}_2$ and thus rule *b.vi* can be applied. ▮

LEMMA 14. *If there exists some reduction rule in $\mathscr{R}_{b \leq 3}$ occurring in a graph G, then, for any subgraph G' of G such that $V(G') = V(G)$, there exists some rule in $\mathscr{R}_{b \leq 3}$ occurring in G' as well.*

Proof. It is sufficient to prove that if some reduction rule R occurs in a graph G then for any $e \in E(G)$ there exists some rule in $\mathscr{R}_{b \leq 3}$ occurring in $G' = (V(G), E(G) - \{e\})$. If the removal of e does not harm the occurrence of R, then R occurs in G' as well. If this is not the case we claim that, whatever the rule R is, the removal of e implies the occurrence of another rule $R' \in \mathscr{R}_{b \leq 3}$ in G'. Indeed, it is not difficult to check that any removal of an edge in rule *b.ii*, *b.iii*, *b.iv*, *b.v*, produces rule *b.i*, *b.ii*, *b.iii*, *b.iii*, respectively and any removal of an edge in rule *b.vi*, produces either rule *b.iii* or rule *b.iv*. ▮

LEMMA 15. *$\mathscr{R}_{b \leq 3}$ is a complete set of reduction rules for the class of graphs with branchwidth ≤ 3.*

Proof. Let G be a nonempty graph with branchwidth ≤ 3. We will prove that there is a reduction rule in $\mathscr{R}_{b \leq 3}$ occurring in G. From Theorem 5, G has treewidth bounded by 3 and is crossless. Let G' be a minimal triangulation of G. According to Lemma 3, G' is also crossless. From Lemma 7, we have that G is a subgraph of a crossless 3-tree G'' such that $V(G'') = V(G)$. Applying now Lemma 13 we know that there exists a reduction rule in $\mathscr{R}_{b \leq 3}$ occurring in G''. The result now follows immediately from Lemma 14. ▮

Now, from Lemma 11 and 15 we have the following.

THEOREM 7. *$\mathscr{R}_{b \leq 3}$ is a safe and complete set of rules for rewriting graphs of branchwidth at most* 3.

5. THE ALGORITHM

The results of the previous sections lead to the construction of a linear time algorithm testing whether a graph has branchwidth at most 3 and, if so, computes a branch decomposition of minimum width. In this section we present this algorithm.

LEMMA 16. *Let G be a graph with branchwidth at most* 3. *Let also $(R_1, \ldots, R_r)$ be a sequence of reduction rules in $\mathscr{R}_{b\leq 3}$ that reduces G to the empty graph. Then, one can construct an algorithm that, given G and $(R_1, \ldots, R_r)$, outputs in $\mathscr{O}(|V(G)|)$ time a crossless chordal graph G' with maximum clique size at most* 4, *such that G is a subgraph of G' and $V(G') = V(G)$.*

Proof. Let $G_1, \ldots, G_{r+1}$ be a sequence of graphs such that $G = G_1, G_{r+1}$ is the empty graph, and G_{i+1} occurs after the application of R_i to G_i for $i = 1, \ldots, r$. Clearly, we can compute, in $\mathscr{O}(|V(G)|)$ time the set $E_+ = \bigcup_{i=1,\ldots,r} E(G_i)$. It is now easy to see that $G' = (V(G), E_+)$ is the required crossless chordal graph and such a graph can be constructed in $\mathscr{O}(|V(G)|)$ time. ▮

LEMMA 17. *One can construct a linear time algorithm that, given a graph G, checks whether* branchwidth$(G) \leq 3$ *and, if so, outputs, in $\mathscr{O}(|V(G)|)$ time, a sequence $(R_1, \ldots, R_r)$ of reduction rules in $\mathscr{R}_{b\leq 3}$ that can reduce a graph G to the empty graph.*

Proof. According to Theorem 7, a graph has branchwidth at most 3 iff we can find a sequence of reduction rules in $\mathscr{R}_{b\leq 3}$ that, when successively applied, reduce G to the empty graph. Clearly, it is enough to prove that there exists a way to check, in constant time, whether a reduction rule in $\mathscr{R}_{t\leq 3}$ occurs in a graph G or not. Indeed, it is enough to observe that, given a reduction rule $(H, S, f) \in \mathscr{R}_{b\leq 3}$, every edge of H contains a vertex of degree ≤ 5 and then to use the same approach as the one used in [14] by Matoušek and Thomas (their algorithm is based on the same observation about the set $\mathscr{R}_{t\leq 3}$). (See, however, the note added in proof.) ▮

We are now in position to present the main algorithm of this paper.

ALGORITHM. Branchwidth-3.
input: A graph G.
output: A branch decomposition of G with width ≤ 3 or a report that branchwidth$(G) > 3$.

1. Use G and the algorithm of Lemma 17 and create sequence $(R_1, \ldots, R_r)$. If such a sequence does not exist then report that branchwidth$(G) > 3$ and **stop**.

2. Use $G, (T_1, \ldots, R_r)$, and the algorithm of Lemma 16 to construct a crossless chordal graph G' with maximum clique at most 4 that contains G as a subgraph.

3. Use G' and the algorithm of Lemma 7 and construct a crossless k-tree G'' that contains G' (and therefore G) as a subgraph.

4. Use G'' and algorithm Clique-Tree to construct a 3-clique tree $T_{G''}$ of G''.

5. Use $T_{G''}$ and the algorithm of Lemma 8 to construct a labelled 3-tree clique $(T_{G''}, l)$ of G''.

6. Use $(T_{G''}, l)$, and the algorithm of Lemma 9 to construct an amplified branch decomposition B^* of G'' with width 3.

7. Use B^* and the algorithm of the second statement of Lemma 1 to construct a branch decomposition B'' of G'' with width at most 3.

8. Use B'' and the algorithm of the first statement of Lemma 1 to construct a branch decomposition B of G with width at most 3.

9. Output B.

10. **end**

Notice that each of the nine steps of algorithm Branchwidth-3 can be done in $\mathcal{O}(|V(G)|)$ time. Notice also that, according to Theorem 1.d, one can check, in linear time, if a graph has branchwidth at most 2 or not. Moreover, from Theorem 1.c, it is trivial to check, in linear time, if G has branchwidth at most 1 or not. Therefore, it is easy to know if G has branchwidth 2. In this special case, the corresponding branch decomposition can be computed using a straightforward modification of our algorithm. Finally, if branchwidth$(G) = 1$ then, using Theorem 1.c, it is trivial to construct the minimal branch decomposition. Concluding, we sum up the results of this paper as follows.

THEOREM 8. *The following three statements hold*:

a. *A graph has branchwidth at most* 3 *if and only if it does not contain any of the graphs in the set* $\mathcal{B}_3 = \{K_5, M_6, M_8, Q_3\}$ *as a minor*.

b. *A graph has branchwidth at most* 3 *if and only if there exists a sequence of reduction rules in* $\mathcal{R}_{b \leq 3}$ *that can reduce G to the empty graph*.

c. *One can construct an algorithm that tests if a given graph G has branchwidth at most* 3 *and, if so, outputs a branch decomposition of minimum width in* $\mathcal{O}(|V(G)|$ *time*.

6. OPEN PROBLEMS

We believe that the methodology applied in this paper may be useful in identifying obstruction sets and/or reduction rules for other problems as well. In this direction, the study of the graphs with branchwidth at most four appears to be an interesting problem.

ACKNOWLEDGMENT

We thank Kouichi Yamazaki for discussions on this research.

Note Added in Proof. Upon recent inspection of [14], we became uncertain whether the technique used to find occurrences of the rule t.v = b.v (i.e., find pairs of vertices of degree 3 with the same neighbors in a dynamically changing graph) works as required. Using a more than linear space, similarly to techniques used in [2], can in any case resolve the problem, assuming the RAM with uniform cost measure computation model. Thus, while the time needed for the algorithm is linear, the space used is $O(n^3)$. Techniques such as that described in T. Hagerup, On Saving Space in Parallel Computation, *Inform. Process. Lett.* **29** (1988), 327–329, could help to bring the space requirements down, e.g., to $O(n^{1+\epsilon})$. It is also not difficult to see that an implementation exists that uses $O(n \log n)$ time and $O(n)$ space.

REFERENCES

1. S. Arnborg, D. G. Corneil, and A. Proskurowski, Complexity of finding embeddings in a *k*-tree, *SIAM J. Alg. Discrete Methods* **8** (1987), 277–284.
2. S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese, An algebraic theory of graph reduction, *J. Assoc. Comput. Mach.* **40** (1993), 1134–1164.
3. S. Arnborg and A. Proskurowski, Characterization and recognition of partial 3-trees, *SIAM J. Alg. Disc. Meth.* **7** (1986), 305–314.
4. S. Arnborg, A. Proskurowski, and D. G. Corneil, Forbidden minors characterization of partial 3-trees, *Discrete Math.* **80** (1990), 1–19.
5. H. L. Bodlaender, "A Partial *k*-Arboretum of Graphs with Bounded Treewidth," Theor. Comp. Sc. **209** (1998), 1–45.
6. H. L. Bodlaender, T. Kloks, and D. Kratsch, Treewidth and pathwidth of permutation graphs, *SIAM J. Disc. Meth.* **8**, No. 4 (1995), 606–616.
7. H. L. Bodlaender and D. M. Thilikos, Constructive linear-time algorithms for branchwidth, in "Proceedings 24th International Colloquium on Automata, Languages, and Programming" (P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, Eds.), Lecture Notes in Computer Science, Vol. 1256, pp. 627–637, Springer-Verlag, New York/Berlin, 1997.
8. H. L. Bodlaender, J. van Leeuwen, R. Tan, and D. Thilikos, On interval routing schemes and treewidth, *Inform. and Comput.* **139**, No. 1 (1997), 92–109.
9. W. Cook, 1996. Personal communication.
10. W. Cook and P. D. Seymour, An algorithm for the ring-routing problem, Bellcore technical memorandum, Bellcore, 1993.
11. F. Gavril, The intersection graphs of subtrees in trees are exactly the chordal graphs, *J. Combin. Theory Ser. B* **16** (1974), 47–56.

12. Y. Kajitani, A. Ishizuka, and S. Ueno, A characterization of the partial k-tree in terms of certain substructures, *Graphs Combin*. **2** (1986), 233–246.
13. T. Kloks, ''Treewidth. Computations and Approximations,'' Lecture Notes in Computer Science, Vol. 842, Springer-Verlag, Berlin, 1994.
14. J. Matoušek and R. Thomas, Algorithms finding tree-decompositions of graphs, *J. Algorithms* **12** (1991), 1–22.
15. N. Robertson and P. D. Seymour, Graph minors—A survey in ''Surveys in Combinatorics'' (I. Anderson, Ed.), pp. 153–171, Cambridge Univ. Press, Cambridge, 1985.
16. N. Robertson and P. D. Seymour, Graph minors. X. Obstructions to tree-decomposition, *J. Comb. Theory Series B*, **52** (1991), 153–190.
17. D. P. Sanders, On linear recognition of tree-width at most four, *SIAM J. Disc. Meth*., **9**, No. 1 (1996), 101–117.
18. A. Satyanarayana and L. Tung, A characterization of partial 3-trees, *Networks* **20** (1990), 299–322.
19. P. D. Seymour and R. Thomas, Call routing and the ratcatcher, *Combinatorica* **14**, No. 2 (1994), 217–241.
20. J. van Leeuwen, Graph algorithms, in *Handbook of Theoretical Computer Science, A: Algorithms and Complexity Theory*, pp. 527–631, North Holland, Amsterdam, 1990.
21. J. A. Wald and C. J. Colburn, Steiner trees, partial 2-trees, and minimum IFI networks, *Networks* **13** (1983), 159–167.
22. M. Yannakakis, Computing the minimum fill-in is NP-complete, *SIAM J. Alg. Disc. Meth*. **2** (1981), 950–988.
23. M. Yannakakis and F. Gavril, The maximum k-colorable subgraph problem for chordal graphs, *Inform. Proc. Lett*. **24** (1987), 133–137.