Graphs with Branchwidth at most Three*†

Hans L. Bodlaender Dimitrios M. Thilikos‡

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

Abstract

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 of graphs with branchwidth at most three. 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 checks if a given graph has branchwidth at most three, and, if so, outputs a minimum width branch decomposition.

Keywords: graph algorithms, branchwidth, obstruction set, graph minor, reduction rule.

1 Introduction

This paper considers the study of 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.

*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]).

†This research was partially supported by ESPRIT Long Term Research Project 20244 (project ALCOM IT: *Algorithms and Complexity in Information Technology*).

‡The second author was supported by the Training and Mobility of Researchers (TMR) Program, (EU contract no ERBFMBICT950198).

The branchwidth of a graph differs from its treewidth by at most a multiplicative constant factor (see Theorem 1.) As branchwidth is also reflecting 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 starting point for the 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 [9], and remarked that branchwidth was the more natural notion (instead of treewidth) to use for that problem [8]: 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 [18]). As both treewidth and branchwidth are NP-complete parameters (see [1, 18]), it appears an interesting task to find algorithms solving the following problems (k is assumed to be a fixed constant).

$\Pi_k^d(B)$ ($\Pi_k^d(T)$): Check if an input graph has branchwidth (treewidth) at most k.
$\Pi_k^c(B)$ ($\Pi_k^c(T)$): Given a graph with branchwidth (treewidth) at most k, output a minimum width branch (tree) decomposition.

According to the results of Robertson and Seymour, for any minor closed class of graphs there exists a finite set of graphs, its *obstruction set*, such that a graph G belongs in 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 minor closed (see also Theorem 1). An immediate consequence of this fact (using results from Robertson and Seymour and the algorithm from [5]) is the existence of a linear time algorithm solving $\Pi_k^d(B)$ or $\Pi_k^d(T)$. Unfortunately, in this way, we only get a non-constructive 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 non only decides on branchwidth, but also constructs the corresponding branch decomposition.

Much research has been done towards the construction of linear time algorithms solving $\Pi_k^d(T)$ and $\Pi_k^c(T)$. In [5], a linear (on the size of the input) time algorithm for treewidth was constructed. As this algorithm appears to be heavily exponential on k (and thus impractical, at least without considerably optimisations in the implementation), practical "tailor-made" algorithms have been presented for small values of k: (treewidth 1 and 2 [12, 20], treewidth 3 [3, 10, 12], treewidth 4 [16].) Also, the obstruction sets for treewidth 1, 2, and 3 are known [4, 17, 20]. Recently, a linear time algorithm solving $\Pi_k^d(B)$ and $\Pi_k^c(B)$ was given in [7]. Unfortunately, the algorithms in [7] appear (similarly to the case of treewidth) to be non-practical.

In this paper, we provide special "tailor made" results for the case where $k \leq 3$. More specifically, for the class of graphs with branchwidth ≤ 3, 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 ≤ 3 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 Figure 2 and the proof of its correctness is based on a structural lemma asserting that the graphs of branchwidth≤ 3 are exactly the graphs that with treewidth≤ 3 that do not contain the three-dimensional binary cube graph (i.e. graph Q_3 of Figure 2) as a minor.

The paper is organised 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 of the class of graphs with branchwidth≤ 3. In Section 4, we identify a complete and safe set of reduction rules leading to the construction of a practical linear time algorithm solving $\Pi_3^d(B)$ and $\Pi_3^c(B)$.

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 $V(G)$ 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 as $N_G(v)$ the set of vertices in $V(G)$ adjacent with v. Given a set $S \subseteq V(G)$ we denote as $G[S]$ the graph induced by S. We also denote as K_r the complete graph with r vertices. Finally, we will assume that all the graphs we deal with are connected, as this does not harm 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 $H \preceq G$) 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 where 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 in $\mathcal{G}$. Given a graph class $\mathcal{G}$ that is closed under taking of minors, we define the *obstruction set* of $\mathcal{G}$ as the set set of minor minimal graphs that do not belong in $\mathcal{G}$. Robertson and Seymour proved (see e.g. [14]) 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 H, $G \in \mathcal{G}$ iff there is no graph in the obstruction set of $\mathcal{G}$ such that $H \preceq G$.

We give now the formal definitions of treewidth and branchwidth.

A *tree decomposition* of a graph G is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, where $\{X_i \mid i \in I\}$ 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 $\{i \mid v \in X_i\}$ forms a subtree of T.

The *width* of a tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ 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 incident with v (we also say: v belongs to e.)

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 case where $|E(G)| \leq 1$, then 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, 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 leave v of T with $\tau(v) = e$.

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

Theorem 1 ([15]) *The following statements hold.*

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

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

c. The graphs with branchwidth at most 0 *(at most* 1*) are all graphs where each connected component contains at most one edge (vertex of degree at least* 2*).*

d. $\mathcal{B}_2 = \{K_4\}$.

Lemma 1 *There exists an algorithm that given a branch decomposition (T, τ) of a graph G with width at most* 3*, outputs a branch decomposition of any subgraph G' of G with width at most* 3 *in $O(|V(G)|)$ time. Moreover, there exists an algorithm that given an amplified branch decomposition (T, τ) of a graph G with width at most* 3*, outputs a branch decomposition of G with width at most* 3*, in $O(|V(T)|)$ time.*

Proof. In order to prove the first statement of the lemma we set $E_r = E(G')$ and for the second statement we set $E_r = \cup_{e \in E(G)}\{v_e\}$ where v_e is some vertex in V_e and $V_e = \{v \in V(T) : \tau(v) = e\}$.

For both statements of the lemma we set $V^* = \{v \in V(T) : \exists e \in E_r \text{ such that } \tau(e) = v\}$. Let T' be the tree obtained from T as follows: (i) remove leaves that do not belong in V^* until

no such leaves occur any more (ii) contract edges consisting of a vertex of degree 2 and a vertex in V^* until no such edges occur any more. Finally, we define τ' as the restriction of τ on V^*. It is now easy to see that, for both of the statements of the lemma, (T', τ') is the required branch decomposition and can be computed in linear time. □

A *reduction* R is a triple (H, S, f), where H is a graph $S \subseteq V(G), S \neq \emptyset$ 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 $R = (H, S, f)$ *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 $V(H) - S$.

Given a graph class $\mathcal{G}$, we say that a set $\mathcal{R}$ of reductions is *safe* if, for any $R \in \mathcal{R}$ and for any G such that R occurs in G, the result of applying R on G is a graph in $\mathcal{G}$ if and only if $G \in \mathcal{G}$.

$\mathcal{R}$ is called *complete* for $\mathcal{G}$, if for every non-empty graph $G \in \mathcal{G}$, there is a reduction in $\mathcal{R}$ occurring in G.

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

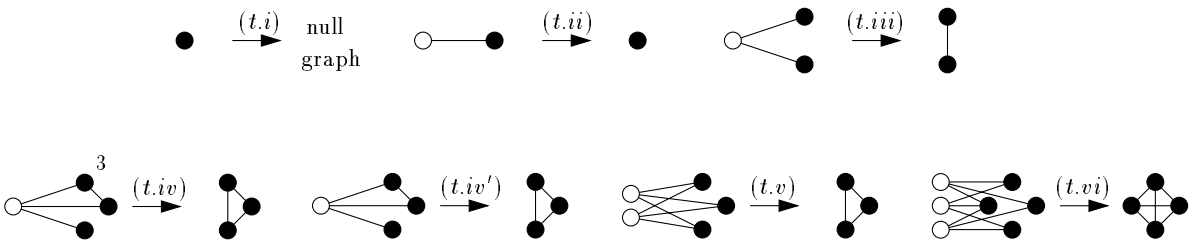

Figure 1: The reduction rules for the class of graphs with treewidth ≤ 3.

We denote as $\mathcal{R}_{t\leq 3}$ the set of reduction rules $\{t.i, t.ii, t.iii, t.iv, t.v, t.vi\}$, shown in Figure 1. For any $R = (H, S, f) \in \mathcal{R}_{t\leq 3}$, S is represented by the white cycles and the values of f are shown only when they are not ω and correspond to vertices not in S.

Theorem 2 ([3, 10, 13]) *$\mathcal{R}_{t\leq 3}$ 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 $\mathcal{R}_{t\leq 3}$ with $t.iv'$ the resulting set of rules is also safe and complete for the class of graphs with treewidth ≤ 3.*

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

We call a graph G *chordal* when it does not contain any induced cycle of length ≥ 4.

We call a vertex $v \in V(G)$ *simplicial* if $G[N_G(v)]$ is a clique.

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 $\leq k$ 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.

Let k be an integer. A k-tree is a graph which is recursively defined as follows. A clique with $k+1$ vertices is a k-tree. Given a k-tree G with n vertices, a k-tree with $n+1$ vertices can be constructed by making a new vertex adjacent to the vertices of a k-clique in G. 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. k-Trees are chordal graphs with maximum clique size $k+1$.

It can be easily proved that a graph has treewidth $\leq k$ iff it is a partial k-tree (see e.g. [19]). 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 $O(kn)$ time.

A set $S \subseteq V(G)$ is an *s-t-separator* in G (s, $t \in V$), 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 subgraph. 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*.

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.*

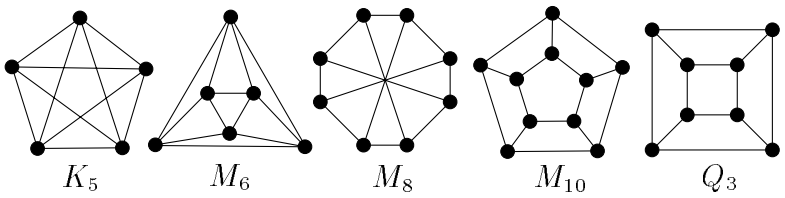

Figure 2: The graphs K_5, M_6, M_8, M_{10}, and Q_3

Graphs K_5, M_6, M_8, and M_{10} are shown in Figure 2.

Theorem 4 ([4, 17]) $\mathcal{T}_3 = \{K_5, M_6, M_8, M_{10}\}$.

Lemma 2 *The following three statements hold.*

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

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

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

Proof. a. From Theorem 1.1, 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 after an vertex/edge deletion or edge contraction has a branch decomposition of width at most 3.
c. We will first prove that $\{K_5, M_6, M_8\} \subseteq \mathcal{B}_3$. From Theorem 4, $\mathcal{S} = \{K_5, M_6, M_8\} \subseteq \mathcal{T}_3$ and thus $\forall G \in \mathcal{S}$: treewidth$(G) = 4$. From Theorem 1.b, we obtain that branchwidth$(G) \geq \lceil \frac{2}{3}(\text{treewidth}(G) + 1) \rceil$ and thus $\forall G \in \mathcal{S}$: branchwidth$(G) \geq 4$. It is now enough to check, by inspection, that if we apply to any element of 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 $\mathcal{B}_3 - \{K_5, M_6, M_8\}$ that has treewidth ≥ 4. If this is the case, G would contain one of the graphs in $\mathcal{T}_3$ as a minor and thus one of the graphs in $\{K_5, M_6, M_8, Q_3\} \subseteq \mathcal{B}_3$ which is a contradiction (observe that $Q_3 \preceq M_{10}$ and $Q_3 \in \mathcal{B}_3$). □

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 the sets $S_i = S - \{v_i\}, 1 \leq i \leq 4$ are all minimal separators of G. We also define as $\mathsf{att}(G, S_i)$ the set of all the vertices of the connected components of $G[V(G) - S_i]$ that do not contain the single vertex in $S - S_i$. 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 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 [11]). What remains to prove is that G' is crossless. Suppose that G' contain a cross S. Then all the triples of S are minimal separators and, because of Theorem 3, they are also minimum separators of G. We now have a contradiction as G is crossless. □

We now introduce the notion of the clique tree of a 3-tree. (We mention that it is possible to extent the definition below – as well as the algorithm following it – for any integer $k \neq 3$.)

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 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 $\{\mathbf{v}, \mathbf{u}\}$ in T_G then $|\mathbf{v} \cap \mathbf{u}| = 3$, i.e., they have exactly 3 vertices in common (notice that each such triple of vertices is a minimal separator of G).

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. From now on we will denote the vertices and the edges of a clique tree using bold characters like $\mathbf{v}, \mathbf{u}, \mathbf{e}$.

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

Algorithm $4CT$
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** $C = \bigcup_{v \in \mathsf{sim}(v_i)} \mathsf{sim}(v)$;
11: find the unique $v_{i'} \in C$ such that $\mathbf{a}_{i'} = \mathsf{sim}(v_i) \cup \{v_{i'}\}$ is a 4-clique;
12: **let** $\mathbf{a}_i = \mathsf{sim}(v_i) \cup \{v_i\}$;
13: **let** $V(T_G) = V(T_G) \cup \{\mathbf{a}_i\}, E(T_G) = E(T_G) \cup \{\{\mathbf{a}_{i'}, \mathbf{a}_i\}\}$;
14: **end**
15: **end**

Lemma 4 *Let $(v_1, \ldots, v_n)$ be a perfect elimination ordering of a 3-tree G and $(G_1, \ldots, G_n)$ the corresponding graph sequence. Let also $\mathsf{sim}(v_i), i = 1, \ldots, n$ be as defined in lines 3–6 of algorithm $4CT$. Then, for any G_i, there exists exactly one vertex $v_{i'} \in C = \cup_{v \in \mathsf{sim}(v_i)} \mathsf{sim}(v)$ such that $G[\{v_{i'}\} \cup \mathsf{sim}(v_i)]$ is a 4-clique.*

Proof. Let $\{v_j, v_{j'}, v_{j''}\} = N_{G_i}(v_i) = \mathsf{sim}(v_i)$. W.l.o.g. we assume that $j < j', j < j''$. Clearly, $\{v_{j'}, v_{j''}\} \in N_{G_j}(v_j) = \mathsf{sim}(v_j) \subseteq C = \cup_{v \in \mathsf{sim}(v_i)} \mathsf{sim}(v)$. Let also $\{v_{i'}\} = N_{G_j}(v_j) - \{v_{j'}, v_{j''}\}$. Notice that $\{v_{i'}, v_j, v_{j'}, v_{j''}\}$ is a 4-clique in G. Suppose also that there exists an other vertex $u \in C, u \neq v_{i'}$ such that $G[\{u\} \cup \mathsf{sim}(v_i)]$ is a 4-clique. Clearly, $u \in V(G_j)$ and u must be adjacent, in G_j, with all the vertices in $\{v_j, v_{j'}, v_{j''}\}$ and hence with v_j, which is a contradiction as $u \notin \mathsf{sim}(v_j) = N_{G_j}(v_j) = \{v_{j'}, v_{j''}, v_{i'}\}$. □

Lemma 5 *Given a 3-tree G, algorithm $4CT$ constructs the clique tree of G in $O(|V(G)|)$ time.*

Proof. From Lemma 4, lines 10 and 11 can be executed in constant time. Therefore, the overall complexity of $4CT$ is $O(|V(G)|)$. Observing 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. □

We omit the proof of the following lemma as it is very simple and does not offer any further evidence to the objectives of this paper. We just mention that the algorithm involved is based on a traversal of the graph using a 3-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 $O(|V(G)|)$ time computes all the triconnected components of G.*

Lemma 7 *One can construct an algorithm that, given a crossless chordal graph G with maximum clique size at most 4, outputs, in $O(|V(G)|)$ time, a crossless 3-tree G' such that G is a subgraph of G' where $V(G') = V(G)$.*

Proof. We will examine the non-trivial case where the maximum clique size of G is 4 (the case where the maximum clique size of G is at most 3 is reduced to the non trivial case if we first add edges in G so that we obtain a 2-tree G' containing G as a subgraph and then add in G' all the edges that connect some of its simplicial vertices with the rest of its vertices).

It is easy to see that, using a 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 it is a minimal separator or not and g_H takes as input a vertex or an edge of H and outputs a 4-clique 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} (as the maximum clique size of G is 4, there must be at least one triconnected component of G that is a crossless 3-tree). Suppose now that

(a) G_i and G_j are two triconnected components of G such that $1 \leq |V(G_i) \cap V(G_j)| \leq 2$ and

(b) one of G_i,G_j, say G_i, is a crossless 3-tree.

It is enough to show that, in constant time, we (i) can add edges 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}. Using (a) and (b) we distinguish the following cases:

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

$$f_{G_{ij}}(e) = \begin{cases} \{a, b, u, v\} & \text{if } e = \{v, u\} \\ f_{G_i}(e) & \text{if } e \in E(G_i) \\ f_{G_j}(e) & \text{if } e \in E(G_j) \end{cases} \qquad g_{G_{ij}}(t) = \begin{cases} 0 & \text{if } \{v, u\} \subseteq t \\ g_{G_i}(t) & \text{if } t \subseteq V(G_i) \\ g_{G_j}(t) & \text{if } t \subseteq V(G_j) \end{cases}$$

Case (ii): $V(G_i) \cap V(G_j) = \{a, b\}$ and G_j is a triangle $t = \{a, b, e\}$. Let $f_{G_i}(\{a, b\}) = \{a, b, c, d\}$. If $g_{G_i}(\{a, b, c\}) = 0$ then set $v = d$, otherwise set $v = c$. Now, construct G_{ij} adding $\{v, e\}$ in $G[V(G_i) \cup V(G_j)]$. One can now easy verify that G_{ij} is a crossless 3-tree. The new functions f_{G_i}, g_{G_j} are defined as in case (i).

Case (iii): $V(G_i) \cap V(G_j) = \{a\}$. Choose v, u such that $\{a, v\} \in V(G_i)$ and $\{a, u\} \in V(G_j)$. If we add $\{v, u\}$ in $G[V(G_i) \cup V(G_j)]$, we obtain a biconnected graph containing the following tree

triconnected components: G_i, G_j and the the graph induced by the triangle $\{v, u, a\}$. In this way case (iii) is reduced to cases (i) and (ii). □

3 Obstructions for graphs with branchwidth at most 3

In this section we will identify the set $\mathcal{B}_3$ and find a complete and safe set of reduction rules for the class of graphs with branchwidth at most 3. Our results 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.

The following lemma defines the notion of the labelled clique tree of a crossless 3-tree.

Lemma 8 *Let T_G be a clique tree of a crossless 3-tree G. Let also, for any $\mathbf{v} \in V(T_G) : E_{\mathbf{v}} = \{\mathbf{e} \in E(T_G) : \mathbf{v}$ is incident to $\mathbf{e}\}$. Then, for each $\mathbf{v} \in V(T_G) : |\{\mathsf{sep}(\mathbf{e}) : \mathbf{e} \in E_{\mathbf{v}}\}| \leq 3$. Moreover, it is possible in $O(n)$ time to compute a labelling 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.*

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

We call a clique tree that is labelled as in Lemma 8 *3-labelled* and we denote it as (T_G, l). Given a labelled clique tree (T_G, l), we define the *span degree* of a vertex $\mathbf{v}$ to be equal to $|\{l(\mathbf{e}) : \mathbf{e} \in E_{\mathbf{v}}\}|$. We also call a leaf $\mathbf{u}$ of T_G that is adjacent to a vertex $\mathbf{v}$ *simple* if $|\{\mathbf{e} \in E_{\mathbf{v}} : l(\mathbf{e}) = l(\{\mathbf{u}, \mathbf{v}\})\}| = 1$.

Lemma 9 *Let (T_G, l) be a labelled clique tree containing at least one edge. Then, one of the following holds:*

(i) There exists at least one non-simple leave.

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

(iii) 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 $L_1 = \cup_{\mathbf{v} \in L_{T_G}} N_{\mathbf{v}}$. As $T_G[V(T_G) - L_0]$ is a tree, L_1 must contain a least one vertex $\mathbf{v}$ such that $|N_{T_G}(\mathbf{v}) - L_0| = 1$. Suppose now that any leave in T_G is simple. Then, we can notice that $|N_{T_G}(\mathbf{v}) \cap L_0|$ is either 1 or 2. In the first case, $N_{T_G}(\mathbf{v}) = 2$ and hence $\mathbf{v}$ has span degree at most 2. In the second case, $N_{T_G}(\mathbf{v}) = 3$ and hence $\mathbf{v}$ has span degree at most 3. □

Lemma 10 *There exists a linear time algorithm that, given a 3-labelled clique tree of a crossless 3-tree G, constructs a branch width decomposition of G of width 3.*

Proof. We will describe a construction that, given a 3-labelled clique tree (T_G, l) of a crossless 3-tree G, outputs an amplified branch decomposition (T', τ) of G that has width 3. Suppose that for some $\mathbf{v} \in V(T_G), E_{\mathbf{v}} = \{\mathbf{e}^1_1, \ldots, \mathbf{e}^1_{r_1}, \mathbf{e}^2_1, \ldots, \mathbf{e}^2_{r_2}, \mathbf{e}^3_1, \ldots, \mathbf{e}^3_{r_3}\}$ where $\forall i, 1 \leq i \leq 3 : \forall j, 1 \leq j \leq r_i : l(e^i_j) = i$. 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

• In case $r_i \geq 2$, then we set

$$\begin{aligned}
T^i_{\mathbf{v}} &= (\{v_0\} \cup \{v^i_1, \ldots, v^i_{r_i-1}\} \bigcup_{1 \leq j \leq r_i} \{V(T^{i,j}_{\mathbf{v}})\}, \\
&\quad \{\{v_0, v^i_1\}, \{v^i_1, v^i_2\}, \ldots, \{v^i_{r_i-2}, v^i_{r_i-1}\}\} \bigcup_{1 \leq j \leq r_i} \{E(T^{i,j}_{\mathbf{v}})\}), \text{ where for } j = 1, \ldots, r_i - 1, \\
T^{i,j}_{\mathbf{v}} &= (\{v^i_j, v^i_{j,1}, v^i_{j,2}, v^i_{j,3}, v^i_{j,4}, v^i_{j,5}, v^i_{j,6}\}, \\
&\quad \{\{v^i_j, v^i_{j,1}\}, \{v^i_{j,1}, v^i_{j,2}\}, \{v^i_{j,2}, v^i_{j,3}\}, \{v^i_{j,1}, v^i_{j,4}\}, \{v^i_{j,2}, v^i_{j,5}\}, \{v^i_{j,3}, v^i_{j,6}\}\}) \text{ and} \\
T^{i,r_i}_{\mathbf{v}} &= (\{v^i_{r_i-1}, v^i_{r_i,1}, v^i_{r_i,2}, v^i_{r_i,3}, v^i_{r_i,4}, v^i_{r_i,5}, v^i_{r_i,6}\}, \\
&\quad \{\{v^i_{r_i-1}, v^i_{r_i,1}\}, \{v^i_{r_i,1}, v^i_{r_i,2}\}, \{v^i_{r_i,2}, v^i_{r_i,3}\}, \{v^i_{r_i,1}, v^i_{r_i,4}\}, \{v^i_{r_i,2}, v^i_{r_i,5}\}, \{v^i_{r_i,3}, v^i_{r_i,6}\}\}).
\end{aligned}$$

• in case $r_i = 1$, then we set

$$\begin{aligned}
T^i_{\mathbf{v}} = T^{i,1}_{\mathbf{v}} &= (\{v_0, v^i_{1,1}, v^i_{1,2}, v^i_{1,3}, v^i_{1,4}, v^i_{1,5}, v^i_{1,6}\}, \\
&\quad \{\{v_0, v^i_{1,1}\}, \{v^i_{1,1}, v^i_{1,2}\}, \{v^i_{1,2}, v^i_{1,3}\}, \{v^i_{1,1}, v^i_{1,4}\}, \{v^i_{1,2}, v^i_{1,5}\}, \{v^i_{1,3}, v^i_{1,6}\}\}).
\end{aligned}$$

• In case $r_i = 0$, then we set

$$T^i_{\mathbf{v}} = (\{v_0, v^i_1, v^i_2, v^i_3, v^i_4, v^i_5\}, \{\{v_0, v^i_1\}, \{v^i_1, v^i_2\}, \{v^i_2, v^i_3\}, \{v^i_1, v^i_4\}, \{v^i_2, v^i_5\}\}).$$

Observe that, according to the construction above, edge $\mathbf{e}^i_j$ corresponds to the tree $T^{i,j}_{\mathbf{v}}$ for $1 \leq j \leq r_i, 1 \leq i \leq 3$. For an example illustrating the three cases above, see Figure 3. In the clique tree of Figure 3 vertex v is incident to edges $e^1_1, e^1_2, e^1_3, e^1_4, e^2_1$ where $l(e^1_j) = 1, 1 \leq j \leq 4$ and $l(e^2_1) = 2$. In $T_{\mathbf{v}}$, the subtrees corresponding to the edges labelled with 1 are $T^{1,1}_{\mathbf{v}}, T^{1,2}_{\mathbf{v}}, T^{1,3}_{\mathbf{v}}$, and $T^{1,4}_{\mathbf{v}}$. As the unique edge labelled with 2 is e^2_1, the subtree corresponding to it is $T^2_{\mathbf{v}} = T^{2,1}_{\mathbf{v}}$. Finally, the fact that there are no edges labelled with 3 implies the existence of subtree $T^3_{\mathbf{v}}$ in $T_{\mathbf{v}}$.

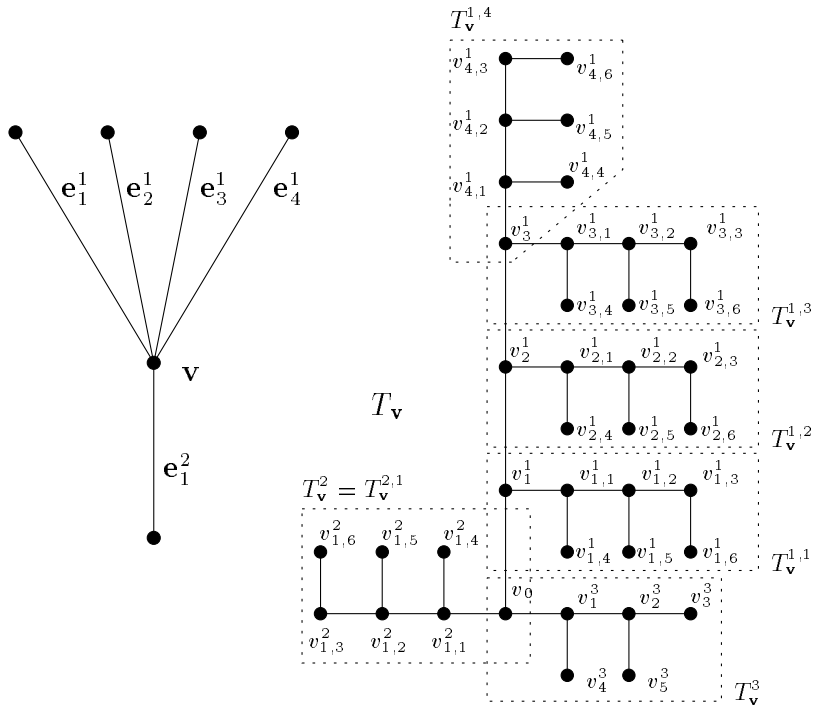

Figure 3: A vertex $\mathbf{v}$ in a clique tree, and the corresponding tree T_v.

The construction of the tree T' of the amplified branch decomposition is now completed as follows: Suppose that two vertices $\mathbf{v}, \mathbf{u}$ in T_G are connected by an edge $\mathbf{e}$. Let also $T_{\mathbf{v}}$ and $T_{\mathbf{u}}$ be the trees corresponding to $\mathbf{v}$ and $\mathbf{u}$ according to the construction above. Moreover, we denote as $T_{\mathbf{v}}^{i_{\mathbf{v}},j_{\mathbf{v}}}$ the subtree of $T_{\mathbf{v}}$ corresponding to edge $\mathbf{e}$ and as $T_{\mathbf{u}}^{i_{\mathbf{u}},j_{\mathbf{u}}}$ the subtree of $T_{\mathbf{u}}$ corresponding to edge e. The construction proceeds by identifying the following couples 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 appear. If we apply this identification for all edges in T_G, we obtain a tree T' with vertices that have degree 1 or 3 which is the tree of the required amplified branch decomposition of G.

What now remains is to define the function τ from the leaves of T' to $E(G)$. There are two kinds of leaves in T'. We first define τ for the leaves appearing in triples of the form $v_{j,4}^i, v_{j,5}^i, v_{j,6}^i$ (we call these leaves *internal*). Notice 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 $\tau(v_{j,6}^i) = \{w_1, w_3\}$.

We now can note that the order of any edge in any $T_{\mathbf{v}}^{i,j}$-type subtree in T' cannot be more than the cardinality of the separating set of the corresponding edge and thus it is equal to 3.

We observe that, so far, for any edge e connecting vertices in some minimal separator of G, there is at least one internal leaf v in T' such that $\tau(v) = e$.

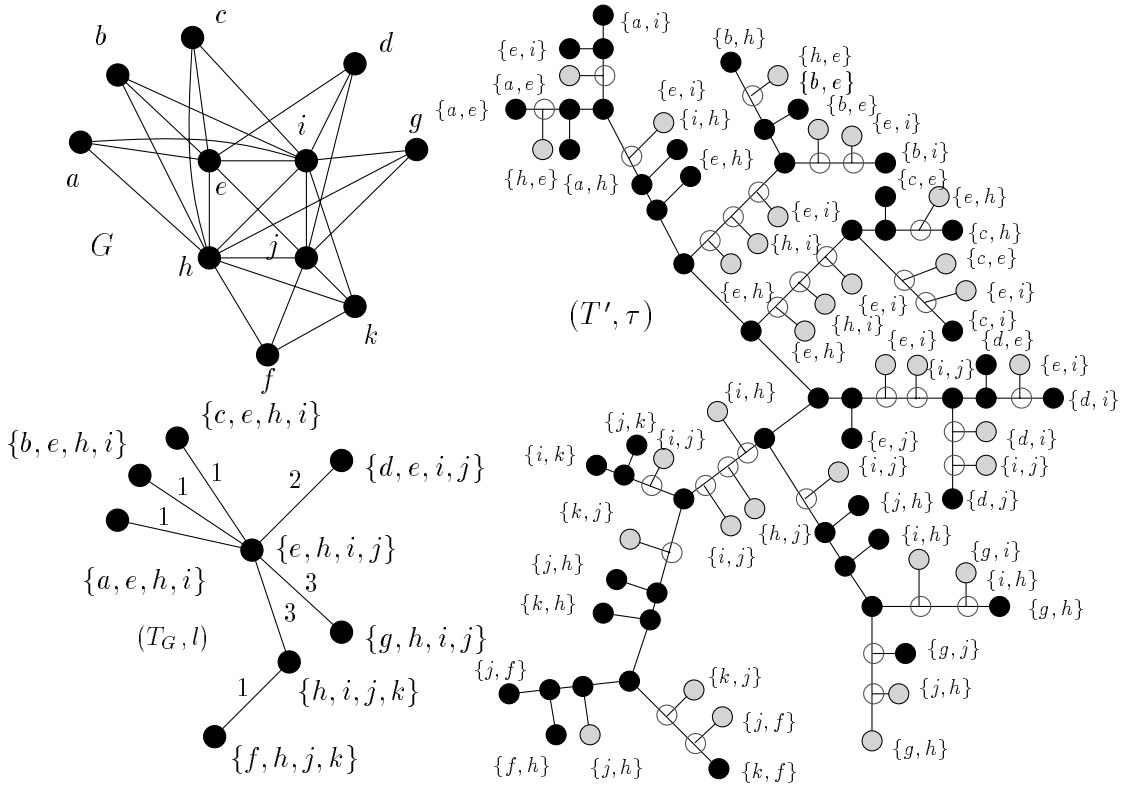

Figure 4: A 3-tree G, a labelled clique tree (T_G, l) of G, and a corresponding amplified branch decomposition (T, τ).

According to the construction above the only leaves of T' for which τ is still undefined are leaves appearing as triples of the form v_3^i, v_4^i, v_5^i (we call these leaves *external*). Notice also that these triples belong to T_v^i-type subtrees of T' and correspond to vertices $\mathbf{v}$ of the clique tree of G that are 4-cliques with the following property: they contain at least two triples that are not a minimal separators of G. Suppose that $\mathbf{v} = \{w_1, w_2, w_3, w_4\}$ is a clique of this type. We distinguish the following cases:

Case (i). $\{w_1, w_2, w_3, w_4\}$ contains exactly 2 triples t_1, t_2 that are minimal separators of G. In this case we observe that it is possible to choose a triple t_3 of $\{w_1, w_2, w_3, w_4\}$ that is not a minimal separator and such that $E(G[t_1 \cup t_2]) \cup E(G[t_3]) = E(G[\{w_1, w_2, w_3, w_4\}])$. If $t_3 = \{w_{i_1}, w_{i_2}, w_{i_3}\}$, then we define $\tau(v_3^i) = \{w_{i_1}, w_{i_2}\}, \tau(v_4^i) = \{w_{i_2}, w_{i_3}\}$, and $\tau(v_5^i) = \{w_{i_1}, w_{i_3}\}$. Observe now that all edges in $T_{\mathbf{v}}^i$ have order 3.

Case (ii). $\{w_1, w_2, w_3, w_4\}$ contains exactly one triple t_1 that is minimal separator of G. In this case we observe that $\mathbf{v}$ is associated with another $T_{\mathbf{v}}^{i'}$-type subtree containing one triple of leaves of the form $v_3^{i'}, v_4^{i'}, v_5^{i'}$ where $i \neq i'$. It is easy now to see that we can find two triples t_2, t_3 in $\{w_1, w_2, w_3, w_4\}$ such that $t_2 \neq t_3$ and $E(G[t_1]) \cup E(G[t_2 \cup t_3]) = E(G[\{w_1, w_2, w_3, w_4\}])$.

If $t_2 = \{w_{i_1}, w_{i_2}, w_{i_3}\}$ and $t_3 = \{w_{i'_1}, w_{i'_2}, w_{i'_3}\}$ then we define $\tau(v_3^i) = \{w_{i_1}, w_{i_2}\}, \tau(v_4^i) = \{w_{i_2}, w_{i_3}\}, \tau(v_5^i) = \{w_{i_1}, w_{i_3}\}, \tau(v_3^{i'}) = \{w_{i'_1}, w_{i'_2}\}, \tau(v_4^{i'}) = \{w_{i'_2}, w_{i'_3}\}$, and $\tau(v_5^{i'}) = \{w_{i'_1}, w_{i'_3}\}$. Notice now that all edges in $T_{\mathbf{v}}^i$ or in $T_{\mathbf{v}}^{i'}$ have order 3.

Case (iii). The case where $\{w_1, w_2, w_3, w_4\}$ does not contain minimal separators is trivial as this can happen only if G is a 4-clique.

We can now see that for any edge e in G whose endpoints belong in a triple that is not a minimal separator of G there exists an external leaf v in T' such that $\tau(v) = e$.

The construction above builds an amplified branch decomposition (T', τ) of G of width exactly 3. It is also not hard to see that it can be easily implemented in $O(|V(G)|)$ time.

Now, according to Lemma 1 it is possible to build a branch decomposition of G in $O(|V(G)|)$ time. An example of the construction we described can be seen in Figure 4. The grey vertices in the tree of the amplified tree decomposition represent the leaves that have to be eliminated in order to obtain the branch decomposition. □

Theorem 5 *One can construct an algorithm that given a crossless chordal graph with maximum clique size at most* 4*, finds a minimum width branch decomposition in* $O(|V(G)|)$ *time.*

Proof. According to Theorem 1.d, we can check in linear time if a graph has branchwidth at most 2 or not. Therefore we can check in linear time if branchwidth$(G) = 3$. In such a case, using Theorem 7, we can construct a crossless k-tree G' containing G as a subgraph and such that $V(G') = V(G)$. Now, using algorithm $4CT$ and Theorems 8, and 10 we can construct a branch decomposition of G' with width 3. Finally, from Lemma 1, we have the required branch decomposition. 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, from Theorem 1.c, it is trivial to construct the minimal branch decomposition. □

We can now proof the following.

Theorem 6 *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 $\Rightarrow$ b). Suppose that branchwidth$(G) \leq 3$. Then, from Theorem 1.b, we get that treewidth$(G) \leq 3$. We also have that $Q_3 \not\leq G$ because, otherwise, from Theorem 1.a, we have that branchwidth$(G) \geq$ branchwidth$(Q_3) = 4$ and this is a contradiction.

(b $\Rightarrow$ c). It is enough to prove that if a graph G has a cross, then it contains $Q_3 \leq G$. Let $S = \{v_1, v_2, v_3, v_4\}$ be a cross in G. We set $S_i = S - \{v_i\}, 1 \leq i \leq 4$. It is now easy to see that if we contract all edges of G with both endpoints in $\bigcup_{i=1,\ldots,4} \mathsf{att}(G, S_i)$, we obtain Q_3.
(c $\Rightarrow$ a). Follows immediately from Lemma 3 and Theorem 5. □

Theorem 7 *The obstruction set of the class of graphs with branchwidth at most three, $\mathcal{B}_3$ equals $\{K_5, M_6, M_8, Q_3\}$.*

Proof. From Lemma 2, it is enough to prove that Q_3 is the only element of $\mathcal{B}_3$ with treewidth equal to 3. This is true because according to Theorem 6, any graph of treewidth at most 3, without containing Q_3 as a minor, has branchwidth at most 3. □

4 Reduction rules for graphs with branchwidth at most 3

We denote as $\mathcal{R}_{b\leq 3}$ the set of reduction rules shown in Figure 5.

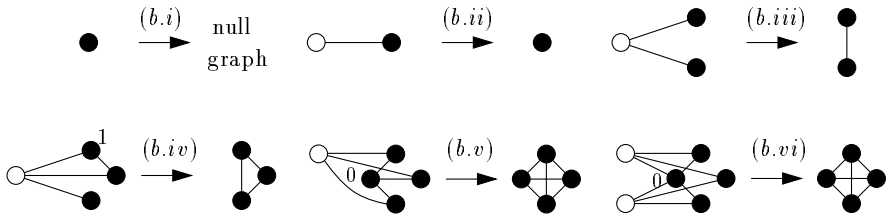

Figure 5: The reduction rules for the class of graphs with branchwidth ≤ 3.

Lemma 11 *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$(G') > 3$. Let $\{a, b, c, d\}$ be the resulting clique in 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, as 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 G'. Now, from Theorem 6, 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 tree 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 in 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 $G'' \preceq G$. 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 with both b,c. Therefore, the occurring graph $G'' = G'[V(G'), E(G') - \{\{b,c\}\}]$ contains a cross which is a contradiction as $G'' \preceq G$. □

Lemma 12 *$\mathcal{R}_{b\leq 3}$ is a safe set of reduction rules for the class of graphs with bounded branchwidth.*

Proof. We have to prove that the application of any reduction rule from $\mathcal{R}_{b\leq 3}$ to a graph preserves both its membership and non-membership. Membership for *b.v* and *b.vi* holds immediately from Lemma 11. For the rest of the rules in $\mathcal{R}_{b\leq 3}$ 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 $H \preceq G$.

Suppose now in contrary, that there exists a graph G with branchwidth ≥ 4 and a reduction rule $R \in \mathcal{R}_{b\leq 3}$ 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 6, G' is crossless and has treewidth ≤ 3. We also have, from the same Lemma, that either G has treewidth ≥ 4 or contains a cross. Suppose that G has treewidth ≥ 4. Notice that, as $\mathcal{R}_{t\leq 3}$ is safe, there are no rules in $\mathcal{R}_{t\leq 3}$ 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. As S is not a cross in G', we assume w.l.o.g. that $\{b,c,d\}$ is not a minimal separator of 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 with b,c and d. This case leads to a contradiction because b,c and c are all adjacent with more than 1 vertices in $V(G'[V(G') - \{b,d,c\}])$ and rule *b.iv* cannot be applied. In the second case G contains two vertices more than G', each one adjacent with different triples in $\{a,b,c,d\}$. In this case we have again a contradiction because a,b,c,d are all adjacent with more than 0 vertices in $G'[V(G') - \{a,b,d,c\}]$ and rule *b.vi* cannot be applied. □

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

Proof. Let (T_G, l) be a labelled clique tree of G. Using Lemma 9, we distinguish the following cases:

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 with the 3 vertices in $\mathsf{sep}(\{\mathbf{v}, \mathbf{u}_1\}) = \mathsf{sep}(\{\mathbf{v}, \mathbf{u}_2\}$ and thus rule *b.vi* can be applied.

Case (ii). Case (i) is excluded and T_G contains a simple leaf $\mathbf{u}$ adjacent to a vertex with span degree d where $d = 1$ or 2. As the case where $d = 1$ is directly covered by the analysis of case (ii), we examine the case where $d = 2$. In this case, we may 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 with the single vertex in $\mathbf{u} - \mathbf{v}$. Observe that the vertex in $\mathbf{v} - S_2$ is adjacent with exactly one vertex not in S_1 and thus rule *b.iv* can be applied.

Case (iii). Cases (i) and (ii) are excluded. In the remaining case there exist two simple leaves $\mathbf{u}_1, \mathbf{u}_2$ in T connected with the same vertex $\mathbf{v}$ and $\mathbf{v}$ has span degree equal to d where $d = 2, 3$. As the case where $d = 2$ is directly covered by the analysis of case (ii), we examine the case where $d = 3$. 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 the single vertex in $\mathbf{v} - S_3$ is adjacent only with vertices in $\mathbf{v}$ and thus rule *b.vi* can be applied. □

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

Proof. It is enough to prove that if some reduction rule R occures in a graph G then, for any $e \in E(G)$, there exist some rule in $\mathcal{R}_{b\leq 3}$ occurring in $G' = (V(G), E(G) - \{e\})$. If the removal of e does not harm the occurence of R, then R occures in G' as well. If this is not the case, then we claim that, whatever the rule R is, the removal of e implies the occurrence of another rule $R' \in \mathcal{R}_{b\leq 3}$ 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 *$\mathcal{R}_{b\leq 3}$ is a complete set of reduction rules for the class of graph with branchwidth ≤ 3.*

Proof. Let G be a non-empty graph with branchwidth ≤ 3. We will prove that there is a reduction rule in $\mathcal{R}_{b\leq 3}$ occurring in G. From Theorem 6, G has bounded treewidth and is crossless. Let G' be a minimal triangulation of G. According to Lemma 3, G' is also crossless. Also, from Lemma 7 G is a subgraph of a crossless 3-tree G'' such that $V(G'') = V(G)$. From Lemma 13 we know that there exists a reduction rule in $\mathcal{R}_{b\leq 3}$ occurring in G''. The result now follows immediately from Lemma 14. □

Now, from Lemmata 12 and 15 we have the following.

Theorem 8 *$\mathcal{R}_{b\leq 3}$ is a safe and complete set of rules for rewriting graphs of branchwidth at most 3.*

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 $\mathcal{R}_{b\leq 3}$ that can reduce a graph G to the empty graph (such a sequence exists because of Theorem 8). Then, one can construct a linear time algorithm that, given G and $R_1, \ldots, R_r$, outputs a crossless chordal graph G' with maximum clique size at most 4 and such that G is a subgraph of G' and $V(G') = V(G)$.*

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

Lemma 17 *One can construct a linear time algorithm that, given a graph G, checks whether branchwidth$(G) \leq 3$ and, if so, outputs a crossless chordal graph G' with maximum clique size at most 4 and such that G is a subgraph of G' and $V(G') = V(G)$.*

Proof. According to Theorem 8, a graph has branchwidth at most 3 iff we can find a sequence of reduction rules in $\mathcal{R}_{b\leq 3}$ that, when successively applied, reduce G to the empty graph. The reductions can be applied in linear time observing that every edge of an occurrence of a reduction of the set $\mathcal{R}_{b\leq 3}$ in a graph G is incident to a vertex of degree ≤ 5 and using the same approach as the one used in [13] by Matoušek and Thomas (their algorithm is based on the same observation about the set $\mathcal{R}_{t\leq 3}$). Given the sequence of the reduction rules that reduce G to the empty graph, is it now possible, using Lemma 16, to generate in $O(|V(G)|)$ time a crossless chordal graph G' with maximum clique size at most 4 and such that G' is a subgraph of G with $V(G') = V(G)$. □

Now, using Lemmata 1 and 17 and Theorem 5, we can conclude that one can construct an algorithm testing if a given graph has branchwidth at most 3 and, if so, outputs a branchwidth decomposition of width at most 3 in $O(n)$ time. So the main conclusions of this section are summed up by the following.

Theorem 9 *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 has branchwidth at most 3 and, if so, outputs a branch decomposition of minimum width in $O(n)$ time.

5 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.

Acknowledgements

We thank Kouichi Yamazaki for discussions on this research.

References

[1] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Alg. Disc. Meth.*, 8:277–284, 1987.

[2] S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. *J. ACM*, 40:1134–1164, 1993.

[3] S. Arnborg and A. Proskurowski. Characterization and recognition of partial 3-trees. *SIAM J. Alg. Disc. Meth.*, 7:305–314, 1986.

[4] S. Arnborg, A. Proskurowski, and D. G. Corneil. Forbidden minors characterization of partial 3-trees. *Disc. Math.*, 80:1–19, 1990.

[5] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25:1305–1317, 1996.

[6] H. L. Bodlaender, T. Kloks, and D. Kratsch. Treewidth and pathwidth of permutation graphs. *SIAM J. Disc. Meth.*, 8(4):606–616, 1995.

[7] H. L. Bodlaender and D. M. Thilikos. Constructive linear time algorithms for branchwidth. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Proceedings 24th International Colloquium on Automata, Languages, and Programming*, pages 627–637. Springer Verlag, Lecture Notes in Computer Science, vol. 1256, 1997.

[8] W. Cook, 1996. Personal communication.

[9] W. Cook and P. D. Seymour. An algorithm for the ring-routing problem. Bellcore technical memorandum, Bellcore, 1993.

[10] Y. Kajitani, A. Ishizuka, and S. Ueno. A characterization of the partial k-tree in terms of certain substructures. *Graphs and Combinatorics*, 2:233–246, 1986.

[11] T. Kloks. *Treewidth*. PhD thesis, Utrecht University, Utrecht, the Netherlands, 1993.

[12] J. Matoušek and R. Thomas. Algorithms finding tree-decompositions of graphs. *J. Algorithms*, 12:1–22, 1991.

[13] J. Matoušek and R. Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Disc. Math.*, 108:343–364, 1992.

[14] N. Robertson and P. D. Seymour. Graph minors — a survey. In I. Anderson, editor, *Surveys in Combinatorics*, pages 153–171. Cambridge Univ. Press, 1985.

[15] N. Robertson and P. D. Seymour. Graph minors. X. Obstructions to tree-decomposition. *J. Comb. Theory Series B*, 52:153–190, 1991.

[16] D. P. Sanders. On linear recognition of tree-width at most four. *SIAM J. Disc. Meth.*, 9(1):101–117, 1996.

[17] A. Satyanarayana and L. Tung. A characterization of partial 3-trees. *Networks*, 20:299–322, 1990.

[18] P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

[19] J. van Leeuwen. Graph algorithms. In *Handbook of Theoretical Computer Science, A: Algorithms and Complexity Theory*, pages 527–631, Amsterdam, 1990. North Holland Publ. Comp.

[20] J. A. Wald and C. J. Colbourn. Steiner trees, partial 2-trees, and minimum IFI networks. *Networks*, 13:159–167, 1983.