Complexity and Exact Algorithms for Vertex Multicut in Interval and Bounded Treewidth Graphs [1]

Jiong Guo [a,*] Falk Hüffner [a] Erhan Kenar [b] Rolf Niedermeier [a] Johannes Uhlmann [a]

[a] *Institut für Informatik, Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany*

[b] *Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Sand 13, D-72076 Tübingen, Germany*

Abstract

MULTICUT is a fundamental network communication and connectivity problem. It is defined as: given an undirected graph and a collection of pairs of terminal vertices, find a minimum set of edges or vertices whose removal disconnects each pair. We mainly focus on the case of removing vertices, where we distinguish between allowing or disallowing the removal of terminal vertices. Complementing and refining previous results from the literature, we provide several NP-completeness and (fixed-parameter) tractability results for restricted classes of graphs such as trees, interval graphs, and graphs of bounded treewidth.

Key words: Combinatorial optimization, Complexity theory, NP-completeness, Dynamic programming, Graph theory, Parameterized complexity

1 Introduction

Motivation and previous results. MULTICUT in graphs is a fundamental network design problem. It models questions concerning the reliability and

* Corresponding author. Tel.: +49 3641 9 46325; fax: +49 3641 9 46002
E-mail address: guo@minet.uni-jena.de.

[1] An extended abstract of this work appears in the proceedings of the 32nd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2006) [19]. Now, in particular the proof of Theorem 5 has been much simplified.

robustness of computer and communication networks. Informally speaking, the problem is, given a graph, to determine a minimum size set of either edges or vertices such that the deletion of this set disconnects a prespecified set of *pairs of terminal vertices* in the graph. In most cases, the problem is NP-complete. There are many results and variants for Multicut and we refer to Costa, Létocart, and Roupin [8] for a recent survey.

The major part of the literature deals with the "edge deletion variant" of Multicut (Edge Multicut) [8,10,16,21,20]. Given a graph and m pairs of terminal vertices, this problem is solvable in polynomial time for $m = 1$ [11] and $m = 2$ [22,23]. For $m \geq 3$, the problem is NP-complete [10]. The problem remains NP-hard and MaxSNP-hard even when the input graph is a star [16].

Our main focus here lies on the "vertex deletion variant" (Vertex Multicut). Relatively little seems to be known for Vertex Multicut problems; we are only aware of two recent investigations [9,26]. Călinescu, Fernandes, and Reed [9] introduced two variants of Vertex Multicut:

Unrestricted Vertex Multicut (UVMC)
Input: An undirected graph $G = (V, E)$, a collection H of pairs of vertices $H \subseteq V \times V$, and an integer $k \geq 0$.
Task: Find a subset V' of V with $|V'| \leq k$ whose removal separates each pair of vertices in H.

The vertices appearing in the vertex pairs in H are called *terminals* and, throughout this paper, we use S to denote the set of terminals, i.e., $S := \bigcup_{(u,v)\in H}\{u, v\}$. By way of contrast, in the case of Restricted Vertex Multicut the removal of terminal vertices is not allowed.

Restricted Vertex Multicut (RVMC)
Input: An undirected graph $G = (V, E)$, a collection H of pairs of vertices $H \subseteq V \times V$, and an integer $k \geq 0$.
Task: Find a subset V' of V with $|V'| \leq k$ that contains no terminal and whose removal separates each pair of vertices in H.

Since RVMC forbids the removal of terminals, there are instances that have no solution, namely when there is a path between a terminal pair in H consisting only of terminal vertices. As it is easy to determine whether or not an instance has a solution for RVMC by checking whether all terminal pairs are disconnected after removing all nonterminal vertices, we assume that all instances under consideration allow for feasible solutions for RVMC.

RVMC is at least as hard as UVMC in general graphs and many special graph classes: From an instance of UVMC we can obtain an "equivalent" RVMC instance by adding for each terminal s a new degree-1 vertex s' adjacent only to s. Each terminal pair (s, t) is substituted by (s', t'). Then, solving RVMC

in this new instance is equivalent to solving UVMC in the original instance.

Călinescu et al. [9] showed that RVMC is NP-complete in bounded-degree trees and the "easier" UVMC is polynomially solvable in trees but becomes NP-complete in bounded-degree graphs of treewidth two. Moreover, they gave a polynomial-time approximation scheme (PTAS) for UVMC in graphs of bounded treewidth. Marx [26] extends the results for UVMC (which he calls Minimum Node Multicut) by providing an $O(2^{k\ell} \cdot 4^{k^3} \cdot |G|^{O(1)})$ time algorithm for UVMC in general graphs, where k is an upper bound on the vertices to be removed and ℓ is the number of terminal pairs. In other words, UVMC is fixed-parameter tractable (FPT) with respect to the combined parameter (k, ℓ). Because of the huge combinatorial explosion in k (and ℓ), however, this result is of mainly theoretical interest and improvements are highly desirable. Finally, we mention in passing that Garg, Vazirani, and Yannakakis [17] studied the vertex deletion variant for the closely related Multiway Cut problem.

Our results—overview. We continue and complement the work of Călinescu et al. [9] and Marx [26] as follows: We show that the NP-complete RVMC in trees is fixed-parameter tractable with respect to the parameter k (number of vertex deletions) with the modest running time $O(|S|^2 \cdot |E| + 2^k \cdot \ell)$ (again, ℓ is the number of terminal pairs). Whereas in trees UVMC is polynomial-time solvable but RVMC is NP-complete [9], we have the surprising result that UVMC is NP-complete in interval graphs but RVMC is polynomial-time solvable here; interval graphs are graphs where the vertices correspond to intervals on the real line and there is an edge corresponding to each pair of intersecting intervals [27]. More specifically, the NP-completeness result for UVMC even holds in interval graphs of pathwidth four. We also strengthen the NP-completeness result for RVMC in trees of Călinescu et al. by showing that NP-completeness already holds for maximum-vertex-degree-three trees whereas their result only holds for maximum vertex degree four. Note that RVMC is clearly polynomial-time solvable in paths, that is, trees with maximum vertex degree two. Moreover, we show that RVMC in general graphs is NP-complete even in case of only three terminal pairs, hence excluding fixed-parameter tractability with respect to the parameter "number of terminal pairs". By way of contrast, we show that RVMC can be solved in $O(|S|^{|S|+\omega+1} \cdot \omega^2 \cdot |V|)$ time on graphs of treewidth ω; thus, RVMC is fixed-parameter tractable with respect to the combined parameter "treewidth" and "terminal set size". Observe that there is no hope for fixed-parameter tractability exclusively with respect to either the parameter $|S|$ or the parameter ω. This fixed-parameter tractability result directly transfers to UVMC as well; indeed, it also works for the Edge Multicut (EMC) variant (note that Bentz [1] independently obtained the EMC fixed-parameter tractability result). On the way to show the NP-completeness of UVMC in interval graphs, we prove that Edge Mul-

Table 1
Computational complexity of MULTICUT problems for several graph classes. For the parameters, $|S|$ is the number of terminals, k is the number of deletions, and ω is the treewidth of the input graph. In a row with a parameter, "NP-c" implies hardness even for some constant parameter value.

Graph class	Parameter	EMC	UVMC	RVMC
Interval graphs		NP-c [16]	NP-c (Thm. 4)	P (Thm. 5)
Trees		NP-c [16]	P [9]	NP-c [9]
	k	FPT [20]	—	FPT (Thm. 2)
General graphs		NP-c [10]	NP-c [9]	NP-c [9]
	k	open	open	open
	$\|S\|$	NP-c [10]	FPT [26]	NP-c (Thm. 6)
	ω	NP-c [16]	NP-c [9]	NP-c [9]
	$\|S\|$ & ω	FPT (Thm. 9, [1])	FPT (Cor. 2)	FPT (Thm. 8)

TICUT is NP-complete in caterpillar graphs with maximum vertex degree five.

Table 1 summarizes most of the presented results.

Preliminaries. We introduce some additional terminology. By default, we consider only undirected graphs $G = (V, E)$ without self-loops. A tree is called *caterpillar* if removing its leaves gives a path. For any graph $G = (V, E)$, we can construct its *line graph* as $(E, \{\{e_1, e_2\} \in E \mid e_1 \cap e_2 \neq \emptyset\})$, that is, the vertices in the line graph one-to-one correspond to the edges in G and two vertices in the line graph are adjacent iff their corresponding edges in G have a common endpoint. For an overview on graph classes, we refer to [6]. We use $G[V']$ to denote the subgraph of G induced by the vertices $V' \subseteq V$, and $V(G)$ to refer to the vertex set V. A set of vertices $V' \subseteq V$ is called *vertex separator* if $G[V \setminus V']$ has more connected components than G. For a minimization problem, a feasible solution is called *minimal* if it does not contain another feasible solution as proper subset, and *minimum* if there is no other feasible solution with better measure.

Many NP-complete graph problems become easy when the input instance is a tree. The notion *treewidth*, introduced by Robertson and Seymour [31], tries to capture the "tree-likeness" of a graph: "tree-like" graphs have small treewidth, and in particular, trees have treewidth one. Many in general NP-hard graph problems can then be solved in polynomial or even linear time when the underlying graph has a treewidth bounded by a constant [2,3,30,32].

Definition 1 *A* tree decomposition *of G is a pair $\langle\{X_i \mid i \in I\}, T\rangle$, where each X_i is a subset of V, called* bag, *and $T = (I, F)$ is a tree with node set I and edge set F. The following must hold:*

(1) $\bigcup_{i \in I} X_i = V$*;*
(2) for every edge $\{u, v\} \in E$*, there is an* $i \in I$ *such that* $\{u, v\} \subseteq X_i$*;*
(3) for all $i, j, l \in I$*, if* j *lies on the path between* i *and* l *in* T*, then* $X_i \cap X_l \subseteq X_j$*.*

The width *of* $\langle \{X_i \mid i \in I\}, T \rangle$ *is* $\max\{|X_i| \mid i \in I\} - 1$*. The* treewidth *of* G *is the minimum width over all tree decompositions of* G*.*

A *path decomposition* is a tree decomposition where T is a path; *pathwidth* is defined analogous to treewidth. For a more detailed introduction to tree decompositions we refer to [2,3,4,25,29,30].

A tree decomposition $\langle \{X_i \mid i \in I\}, T \rangle$ can be transformed in linear time and without affecting its width such that it becomes *nice* [25, Lemma 13.1.3]. Then the following conditions are satisfied:

(1) T is rooted;
(2) every node of the tree T has at most two children;
(3) if a node i has two children j and k, then $X_i = X_j = X_k$ (in this case i is called a *join node*);
(4) if a node i has one child j, then either
   (a) $|X_i| = |X_j| + 1$ and $X_j \subset X_i$ (in this case i is called an *introduce node*), or
   (b) $|X_i| = |X_j| - 1$ and $X_i \subset X_j$ (in this case i is called a *forget node*).

We investigate Multicut problems in the context of parameterized complexity [12,14,29].

A problem is *fixed-parameter tractable* (FPT) with parameter k if an instance of size n can be solved in $f(k) \cdot n^{O(1)}$ time, where f is a computable function solely depending on the parameter k. The idea is to restrict the seemingly unavoidable combinatorial explosion that occurs in exact solutions to NP-hard problems to certain, hopefully small problem parameters.

2 Trees

Unrestricted Vertex Multicut in trees is trivially solvable in $O(|V| \cdot |H|)$ time [9]: Root the tree at an arbitrary vertex. Then, compute the least common ancestors for all terminal pairs and sort these ancestors in a list L (possibly with multiple occurrences of one ancestor) by decreasing order of their depth. [2] Finally, while $L \neq \emptyset$, remove the first element of L and its corresponding vertex

[2] The depth of a vertex in a rooted tree is the length of the path from the root to this vertex.

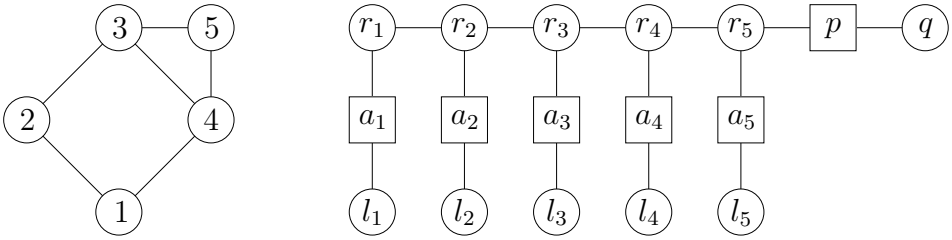

Fig. 1. An example for the reduction from Vertex Cover to RVMC. The left figure is a Vertex Cover instance and the right is the corresponding RVMC instance with $H = \{(l_1, l_2), (l_2, l_3), (l_3, l_4), (l_4, l_1), (l_3, l_5), (l_4, l_5)\} \cup \{(r_i, q) \mid 1 \leq i \leq 5\}$. Only the rectangular vertices can be deleted (all others are terminals). It is easy to see that the graph has a size-3 vertex cover (for example, $\{2, 3, 4\}$), and the RVMC instance has a size-4 solution (for example, $\{p, a_2, a_3, a_4\}$).

from T and delete all separated terminal pairs from H and their least common ancestors from L. The solution is then the set of the removed vertices.

Călinescu et al. [9] showed that RVMC is NP-complete in trees with maximum vertex degree four by giving two reductions, one from 3SAT to EMC in binary trees and one from EMC in binary trees to RVMC. It is easy to observe that RVMC on trees with maximum vertex degree two, that is, paths, can be solved in $O(|V| \cdot |H|)$ time. The complexity of RVMC in trees with maximum vertex degree three remained open. Here we close this gap by giving a direct reduction from Vertex Cover to RVMC.

Theorem 1 Restricted Vertex Multicut *in trees with maximum vertex degree three and pathwidth two is NP-complete.*

PROOF. The reduction is from the NP-complete Vertex Cover problem [15], which for a graph $G = (V = \{v_1, v_2, \ldots, v_n\}, E)$ and $k \geq 0$ asks whether there is a set of vertices $V' \subseteq V$ with $|V'| \leq k$ such that for every edge $\{v, w\} \in E$ at least one of v and w is in V'. Construct the $(3n+2)$-vertices tree $T = (W, F)$ with

$$W := \{l_i, a_i, r_i \mid 1 \leq i \leq n\} \cup \{p, q\}$$

and

$$F := \{\{l_i, a_i\}, \{a_i, r_i\} \mid 1 \leq i \leq n\} \cup \{\{r_i, r_{i+1}\} \mid 1 \leq i < n\} \cup \{\{r_n, p\}, \{p, q\}\}.$$

As the set of terminal pairs H we take for each vertex $v_i \in V$ the pair (r_i, q) and, moreover, for each edge $\{v_i, v_j\} \in E$, we add (l_i, l_j). See Fig. 1 for an example of the construction.

It is easy to show that the Vertex Cover instance has a solution with no more than k vertices iff the constructed RVMC instance can be solved

by removing at most $k+1$ vertices. The constructed tree clearly has maximum vertex degree three and a path decomposition with pathwidth equal to two. □

In the following, we show that RVMC in trees is fixed-parameter tractable with respect to the number of allowed vertex removals k. The basic idea is to modify the polynomial-time algorithm for UVMC in trees [9] into a depth-bounded search tree algorithm. This search tree algorithm is also similar to one for Edge Multicut in trees [20].

Theorem 2 Restricted Vertex Multicut *in trees can be solved in* $O(|S|^2 \cdot |E| + 2^k \cdot |H|)$ *time, where* k *is the number of allowed vertex removals.*

PROOF. Let $T = (V, E)$ be the input instance and $S := \bigcup_{(u,v)\in H}\{u, v\}$ the set of terminals. The first step is to "contract" edges with both endpoints being terminals: For an edge $\{u, v\}$ with $u, v \in S$, we have $(u, v) \notin H$, since otherwise the instance is not solvable. Delete both u and v and the edge between them from T; insert a new vertex w into T and set $N(w) := N(u) \cup N(v) \setminus \{u, v\}$. Furthermore, replace each u and v in H by w. It is easy to see that this step does not change the solution, because neither u nor v could have been taken into a solution.

Then, the search tree algorithm proceeds as the polynomial-time algorithm for UVMC in trees: root T in an arbitrary vertex, compute the least common ancestors of all terminal pairs and sort them by decreasing depth in a list L (possibly with multiple occurrences of one ancestor). While $L \neq \emptyset$, consider the first element u of L, which is the least common ancestor of a terminal pair (v, w). If u is a nonterminal, then remove it and update L and T; otherwise, there are two cases: If $u = v$ or $u = w$, then we delete the neighbor of u that lies on the path from u to w or v. This neighbor has to be a nonterminal due to the first step. Otherwise, we have $u \neq v$ and $u \neq w$. Then u has two nonterminals as neighbors lying on the path between v and w and we branch into two cases, in each case removing one of the two neighbors. This case distinction is sufficient, since choosing any other vertex cannot separate more pairs.

Finally, if there is a node in the search tree where $L = \emptyset$ and at most k vertices have been removed, then we have a solution. It is easy to observe that the depth of the search tree is bounded by k and its size is $O(2^k)$. At each node of the search tree, we update the list L, that is, delete all terminal pairs from H which are no longer connected. Using a prepared list of pairs to mark "dead", this can be done in $O(|H|)$ time. The initial setup takes $O(|S|^2 \cdot |E|$ time. □

3 Interval Graphs

As described in Section 1, RVMC can be easily reduced to RVMC, proving that RVMC is at least as hard as UVMC in general graphs and many special graph classes. However, the class of interval graphs is an exception: UVMC is NP-complete in interval graphs, while RVMC is solvable in polynomial time. This is possible because the described reduction from UVMC to RVMC does not preserve the property of being an interval graph.

3.1 Unrestricted Vertex Multicut in Interval Graphs.

To show the NP-completeness of UVMC in interval graphs, we first show that Edge Multicut is NP-complete in degree-bounded caterpillars and then reduce EMC in caterpillars to UVMC in interval graphs.

Theorem 3 Edge Multicut *in caterpillars with maximum vertex degree five is NP-complete.*

PROOF. We use a reduction from 3-SAT, which is similar to the reduction used to show the NP-completeness of EMC in binary trees [9, Theorem 11].

Let $F(x_1, x_2, \ldots, x_n) = \bigwedge_{j=1}^{m}(l_{j,1} \vee l_{j,2} \vee l_{j,3})$ with $l_{j,k} \in \bigcup_{1 \leq i \leq n}\{x_i, \bar{x}_i\}$ for $1 \leq j \leq m$ and $1 \leq k \leq 3$. We construct a caterpillar tree as follows. First, for every variable x_i, we construct a path consisting of three vertices with the two degree-1 vertices labeled by x_i and $\bar{x}_i$ and add the terminal pair $(x_i, \bar{x}_i)$ to H. We call the path a *variable gadget*.

For each clause $(l_{j,1} \vee l_{j,2} \vee l_{j,3})$, we construct a star with four vertices where the three degree-1 vertices are labeled by $l_{j,1}$, $l_{j,2}$, and $l_{j,3}$, respectively. This star is called *clause star*. The terminal pairs $(l_{j,1}, l_{j,2})$, $(l_{j,1}, l_{j,3})$, and $(l_{j,2}, l_{j,3})$ are added to H.

Then, we add to H the terminal pairs consisting of a degree-1 vertex in the clause stars and the corresponding degree-1 vertex in the variable gadget, that is, if $l_{j,k} = x_i$ (or $l_{j,k} = \bar{x}_i$), then we add the pair $(l_{j,k}, x_i)$ (or $(l_{j,k}, \bar{x}_i)$) to H.

Finally, edges are inserted between the only unlabeled vertices of the clause stars and the variable gadgets in such a way that the resulting graph is a caterpillar. See Fig. 2 for an example.

We claim that the Boolean formula $F(x_1, x_2, \ldots, x_n)$ is satisfiable iff the constructed caterpillar has a solution for EMC of size exactly $n + 2m$, where m

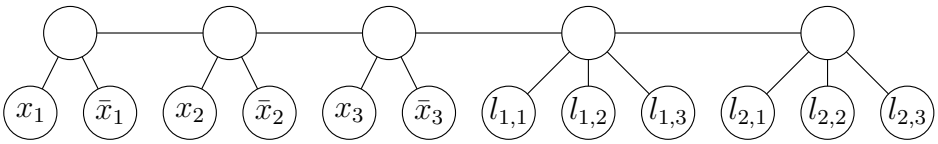

Fig. 2. Example for the transformation of a 3-CNF formula $F(x_1, x_2, x_3) = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \bar{x}_3)$ into a caterpillar as described in the proof of Theorem 3. The terminal pairs for the EMC instance are $H := \{(x_1, \bar{x}_1), (x_2, \bar{x}_2), (x_3, \bar{x}_3), (l_{1,1}, l_{1,2}), (l_{1,1}, l_{1,3}), (l_{1,2}, l_{1,3}), (l_{2,1}, l_{2,2}), (l_{2,1}, l_{2,3}), (l_{2,2}, l_{2,3}), (l_{1,1}, \bar{x}_1), (l_{1,2}, x_2), (l_{1,3}, x_3), (l_{2,1}, x_1), (l_{2,2}, x_2), (l_{2,3}, \bar{x}_3)\}$.

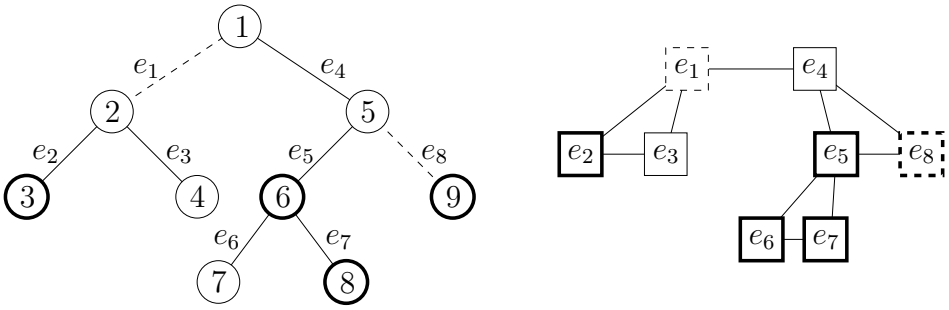

Fig. 3. The figure on the left shows an instance of EMC, where $H = \{(3, 8), (3, 9), (6, 9)\}$. The figure on the right shows the corresponding instance (line graph) for UVMC with $H' = \{(e_2, e_7), (e_2, e_8), (e_5, e_8), (e_6, e_8), (e_7, e_8)\}$. In both instances, removing the set $\{e_1, e_8\}$ yields a minimum solution.

denotes the number of clauses in F. Assuming that every clause consists of exactly three literals, $n + 2m$ is the lower bound for the solution size of EMC on this caterpillar, since, in any case, we have to remove at least one edge from every variable gadget and at least two edges from every clause star. The claim can be proven in a similar way as Lemma 12 in [9]: assigning "true" to x_i corresponds to removing the edge adjacent to x_i, and satisfying a clause by a literal corresponds to not deleting the corresponding edge in the clause star. The size bound effects that an EMC solution corresponds to a valid 3-SAT solution. □

In the second step of our NP-completeness proof, we reduce EMC in caterpillars with maximum vertex degree five to UVMC in interval graphs with pathwidth four. As a rule of thumb, the natural way to reduce an edge-deletion problem to a vertex-deletion problem is to consider line graphs. Analogously to Călinescu et al. [9], we construct the line graph of the caterpillar.

Theorem 4 Unrestricted Vertex Multicut *in interval graphs with pathwidth at least four is NP-complete.*

PROOF. Let $T = (V, E)$ be the input caterpillar and let H be the set of terminal pairs of the EMC instance. We construct the line graph $G = (E, E')$

of T and add for each terminal pair $(u, v) \in H$ all terminal pairs (e, e') to the set of terminal pairs H' of the UVMC instance where the edges e and e' are incident to u and v, respectively. See Fig. 3 for an example. Călinescu et al. showed that then both instances have the same solution [9, Thm. 4], which implies correctness of the above reduction.

It remains to be shown that the pathwidth of the line graph G is four. The line graph of a caterpillar is an interval graph, since the maximal cliques of this line graph can be linearly ordered such that for each vertex v the maximal cliques containing v occur consecutively; this is a well-known characterization of interval graphs [6]. The pathwidth of an interval graph is equal to the maximum clique size minus one, and the maximum clique size in the line graph is equal to the maximum vertex degree of the caterpillar. Thus, the interval graph has pathwidth four. □

3.2 Restricted Vertex Multicut in Interval Graphs

By Theorem 4, UVMC is NP-complete for interval graphs even with bounded pathwidth. This also holds for EMC: EMC is NP-complete even in stars [16], which are interval graphs with pathwidth one. In contrast to this, we now give a polynomial-time algorithm solving RVMC in interval graphs. For an interval graph $G = (V, E)$, it is well-known that an interval representation of G can be computed in $O(|V|+|E|)$ time [5], where the vertices in V one-to-one correspond to the closed intervals of the real line such that two vertices u and v are adjacent if and only if their corresponding intervals have a nonempty intersection. We assume that all intervals have *distinct integral* endpoints [18], that is, the endpoints are labeled by $1, 2, \ldots, 2|V|$ from left to right. In the following, we use the terms "vertex" and "interval" interchangeably.

Our algorithm works on the interval representation. Assume that for each pair $(s, t) \in H$, interval $s = [a, b]$ is to the left of interval $t = [c, d]$, that is, $b < c$. To simplify the presentation, we introduce a point 0 to the left of all points $1, 2, \ldots, 2|V|$. For $0 \leq i \leq 2|V| - 1$, we use G_i to denote the subgraph of G induced by the intervals whose left endpoints are greater than i, S_i to denote the set of terminal intervals contained in G_i, and H_i to denote the set of terminal pairs in H that only contain terminal intervals from S_i. By decreasing order of i, the algorithm computes the optimal solution C_i of the RVMC instance which consists of the graph G_i, the set H_i of terminal pairs, and the terminal set S_i.[3] The computation of C_i, $0 \leq i < 2|V| - 1$, is based on the values of C_j with $i < j \leq 2|V| - 1$. When reaching $G_0 = G$, we have the overall optimal solution C_0 for G. The following pseudo-code shows the

[3] Note that S_i may contain some terminal intervals that do not occur in H_i.

algorithm.

```
1  C_{2|V|-1} ← ∅
2  for i = 2|V| − 2 downto 0:
3      (s, t) ← ([a, b], [c, d]) ∈ H_i such that c is minimum
4      X ← {x | b < x < c and no terminal interval in S_i contains x}
5      for each x ∈ X:
6          Y ← the set of intervals containing x
7          C_i^x ← C_x ∪ Y
8      C_i ← C_i^x with |C_i^x| minimum among all x ∈ X
9  return C_0
```

Observe that for s and t as chosen in line 3 to be separated, there must be some point x with $b < x < c$ that contains no interval. Thus, to separate s and t, we must find some x with $b < x < c$ that is not contained in any terminal interval and then delete all intervals containing x. We collect the candidates for x in line 4. When deleting the intervals in Y for some x, graph G_i splits into two subgraphs: one is induced by the intervals to the left of x, and the other is G_x (recall that G_x contains only intervals with endpoints *greater* than x). Due to the minimality of the left endpoint c of t, there is no terminal pair contained in the first subgraph, and we can ignore this subgraph. Thus, for a choice of x, the optimal solution for G_i is of size $|Y| + |C_x|$, and we obtain C_i by minimizing over all possible values of x (line 8).

Theorem 5 Restricted Vertex Multicut *in interval graphs can be solved in* $O(|V|^2 + |V| \cdot |H|)$ *time.*

PROOF. The correctness of the algorithm directly follows from its description. Concerning the running time of the algorithm, the computation of the interval representation takes $O(|V| + |E|)$ time [5]. For each of the graphs G_i with $0 \leq i \leq 2n - 1$, the algorithm computes the optimal solution C_i by first finding the terminal pair (s, t) where the left endpoint of t is the smallest among all terminal pairs contained in H_i, and, then, considering every point x between the right endpoint of s and the left endpoint of t. Clearly, the computation of C_i needs $O(|V| + |H|)$ time. Therefore, the algorithm runs in $O(|V|^2 + |V| \cdot |H|)$ time. □

The algorithm can be easily extended to handle weighted vertices within the same time bounds.

In the conference version of this paper [19], the proof of Theorem 5 was done in a more complicated way using dynamic programming on path decompositions. The formulation of RVMC in interval graphs as a covering problem we used in [19] is equivalent to the Red-Blue Set Cover problem (introduced

by Carr et al. [7]) restricted to inputs with the so-called consecutive ones property [6,28]. This means that the method described in [19] is also capable of solving this more general problem.

4 General Graphs and Bounded Treewidth

In this section, our main result is a fixed-parameter algorithm for RVMC in general graphs with treewidth and the number of terminals as parameters. All our results here refer to graphs with *given* tree decompositions. Finding such a tree decomposition, however, is generally an NP-hard problem [4].

As shown in Theorem 1, RVMC is NP-complete for tree networks with bounded vertex degree and bounded pathwidth. Therefore, we cannot hope for a fixed-parameter algorithm with only treewidth or pathwidth as parameter. Moreover, in the following theorem we show that RVMC is not fixed-parameter tractable with respect to the number of terminals.

Theorem 6 Restricted Vertex Multicut *is NP-complete if there are at least three terminals.*

PROOF. We give a reduction from EMC to RVMC that preserves the number of terminals and the number of terminal pairs. The theorem follows then from the fact that EMC is NP-complete for at least three terminals [10].

Given an EMC instance with graph $G = (V, E)$, we construct an RVMC instance with graph $G' = (V', E')$. The idea is to keep the terminals, make the nonterminals undeletable by replicating them $(|E| + 1)$-fold, and introduce a vertex w_e as gadget for each edge e. More precisely, with the terminal set S we set

$$V' = S \cup \{u_i \mid 1 \leq i \leq |E| + 1, u \in V \setminus S\} \cup \{w_e \mid e \in E\}.$$

For each edge $e = \{u, v\} \in E$ with $\{u, v\} \subseteq V \setminus S$, we add two edges $\{u_i, w_e\}$ and $\{v_i, w_e\}$ for $1 \leq i \leq |E| + 1$ to E'. For an edge $e = \{u, v\} \in E$ with exactly one of the endpoints being a terminal, say u, add the edge $\{u, w_e\}$ and the edges $\{v_i, w_e\}$ for $1 \leq i \leq |E| + 1$. For an edge $e = \{u, v\} \in E$ with two terminals as endpoints, add two edges $\{u, w_e\}$ and $\{v, w_e\}$. The set of terminal pairs of the RVMC instance is set equal to the set of terminal pairs of the EMC instance. Then, both instances have the same number of terminals.

From a solution of the EMC instance, one can easily construct a solution of the same size for the RVMC instance by choosing w_e for each edge e in the EMC

solution. The mapping can also be reversed, since for any optimal solution C of the constructed RVMC instance with $|C| \leq |E|$, we have $C \subseteq \{w_e \mid e \in E\}$. Therefore, the EMC instance can be solved by l edge removals iff the constructed RVMC instance can be solved by l vertex removals, concluding the proof. □

Because of Theorem 6, we know that there is no hope for a fixed-parameter algorithm for RVMC with respect to the single parameter "treewidth" or the single parameter "number of terminals". In the following, we present a fixed-parameter algorithm for RVMC with treewidth *and* the number of the terminals as parameters. To ease the presentation, we assume that no two terminals are adjacent (two adjacent terminals u, v can be replaced by a new terminal whose neighborhood is $N(u) \cup N(v) \setminus \{u, v\}$).

The basic idea of the fixed-parameter algorithm comes from the fact that any solution of RVMC divides the input graph into at least two connected components such that any two terminals of an input terminal pair are not in the same connected component. Based on this fact, the algorithm consists of two phases. The first phase enumerates all possible partitions of the terminal set that separate all input terminal pairs. It is easy to observe that there are at most $O(|S|^{|S|})$ partitions of the terminal set S.[4] To check whether a partition separates the given terminal pairs in H can be done in $O(|H|)$ time. Then, the running time of the first phase is $O(|S|^{|S|} \cdot |H|)$.

The second phase of the algorithm, for each partition, uses dynamic programming on the tree decomposition to compute the minimum number of vertex removals dividing the input graph into connected components such that each set in this partition is contained in a connected component and no two sets are contained in the same connected component. To simplify the presentation, we give an equivalent formulation of the task of the second phase:[5]

Coloring Extension
Input: An undirected graph $G = (V, E)$, a set of terminals $S \subseteq V$, and a coloring $L_S : S \to C$ with the colors from a set C where $|C| \leq |S|$.
Task: Find an extension $L_G : V \to C \cup \{r\}$ of L_s where $r \notin C$ such that
(1) for every $s \in S$, $L_G(s) = L_S(s)$,
(2) for every edge $\{u, v\} \in E$, either $L_G(u) = L_G(v)$ or $L_G(u) = r$ or $L_G(v) = r$, and
(3) the cost $|\{v \in V \mid L_G(v) = r\}|$ is minimized.

Consider a fixed partition of the terminal set such that for each terminal

[4] More precisely, this is actually the Bell number $B(|S|)$.
[5] A similar coloring problem is defined by Erdős and Székely [13]. Note that here we have a different cost function.

pair its two elements are in different sets of the partition. If we assign to all terminals in one set of this partition the same unique color from C, then a solution of the Coloring Extension problem ensures that every path between two terminals with different colors has to pass through at least one vertex v with $L_G(v) = r$. This implies that the removal of the vertices with color r separates the sets of the partition, which is a solution for the RVMC problem. Minimizing the number of r-colored vertices ensures an optimal RVMC solution.

Theorem 7 *Given an undirected graph $G = (V, E)$ with a tree decomposition of width ω,* Coloring Extension *with the terminal set $S \subseteq V$ colored by the color set C can be solved in $O((|C| + 1)^{\omega+1} \cdot \omega^2 \cdot |V|))$ time.*

PROOF. The algorithm works—without loss of generality—on a nice tree decomposition $\langle X := \{X_i \mid i \in I\}, T = (I, F)\rangle$. Let T_i denote the subtree of T rooted at node i and let $G_i = (V_i, E_i) := G[\bigcup_{j \in T_i} X_j]$. Let $\alpha := |C|$ and let r denote the special color. The coloring of S is given by the function L_S. We seek for an extension of the coloring L_S such that the number of the vertices with color r is minimized.

For each bag $X_i := \{x_{i_1}, \ldots, x_{i_{n_i}}\}$ in X, we use a vector $f \in \{c_1, c_2, \ldots, c_\alpha, r\}^{n_i}$ to denote a coloring of bag X_i, and the color assigned to vertex x_{i_j} by the coloring f is denoted by $f(x_{i_j})$ for $1 \leq j \leq n_i$. Moreover, we define a mapping

$$A_i : \{c_1, c_2, \ldots, c_\alpha, r\}^{n_i} \to \mathbb{N} \cup \{\infty\}.$$

For a coloring $f \in \{c_1, c_2, \ldots, c_\alpha, r\}^{n_i}$, the mapping $A_i(f)$ denotes the minimum cost (that is, number of vertices with color r) over all extensions of the coloring L_S on the subgraph G_i where X_i is colored according to f.

To store the mappings, for each bag i, we allocate a table A_i with $(\alpha + 1)^{n_i}$ rows. We calculate the mappings bottom-up from the leaves of T to the root. First the mappings of the leaves are computed as follows: For all $f = (f_1, f_2, \ldots, f_{n_i}) \in \{c_1, c_2, \ldots, c_\alpha, r\}^{n_i}$, we set

$$A_i(f) := \begin{cases} \infty & \text{if } \exists\, u \in (S \cap X_i) : f(u) \neq L_S(u), \\ \infty & \text{if } \exists\, \{u, v\} \in E_i : \\ & \quad f(u) \neq f(v) \wedge f(u) \neq r \wedge f(v) \neq r, \\ |\{v \in V_i \mid f(v) = r\}| & \text{otherwise.} \end{cases}$$

Since for a leaf i we have $V(G_i) = X_i$, this accounts for all r-colored vertices in G_i, while excluding invalid colorings. The mappings of internal nodes of T

are completely determined by the mappings of their children and the edges in the bag. We distinguish three cases.

Case 1: Forget nodes.
Let i be a forget node with child j. More precisely, $X_i = \{x_{i_1}, \ldots, x_{i_{n_i}}\}$ and $X_j = \{x_{i_1}, \ldots, x_{i_{n_i}}, x\}$. For all $f \in \{c_1, c_2, \ldots, c_\alpha, r\}^{n_i}$, we set

$$A_i(f) := \min\{\, A_j(f \times c) \mid c \in \{c_1, c_2, \ldots, c_\alpha, r\}\}.$$

Note that $f \times c \in \{c_1, c_2, \ldots, c_\alpha, r\}^{n_i+1}$. Since $G_i = G_j$, the correctness trivially follows.

Case 2: Introduce nodes.
Let i be an introduce node with child j. More precisely, $X_j = \{x_{i_1}, \ldots, x_{i_{n_j}}\}$ and $X_i = \{x_{i_1}, \ldots, x_{i_{n_j}}, x\}$. For all $f \in \{c_1, c_2, \ldots, c_\alpha, r\}^{n_i}$ and all $c \in \{c_1, c_2, \ldots, c_\alpha, r\}$, we set

$$A_i(f \times c) := \begin{cases} \infty & \text{if } x \in S \wedge L_S(x) \neq c, \\ \infty & \text{if } \exists\{x, v\} \in E_i : f(x) \neq f(v) \wedge c \neq r \wedge f(v) \neq r, \\ A_j(f) + 1 & \text{if } c = r, \\ A_j(f) & \text{otherwise.} \end{cases}$$

Here G_i is G_j extended by the vertex x, that is $G_i = (V_i, E_i)$ with $V_i = V_j \cup \{x\}$ and $E_i = E_j \cup \{\{x, v\} \in E : v \in V_i\}$. A coloring candidate $f \times c$ is infeasible when it violates the terminal coloring or colors the two endpoints of a newly introduced edge differently without coloring one endpoint by r. Otherwise, the cost increases when the added vertex is an r-vertex, or stays the same when it gets a color from C.

Case 3: Join nodes.
Let i be a join node with children j_1 and j_2. More precisely, $X_i = X_{j_1} = X_{j_2} = \{x_{i_1}, \ldots, x_{i_{n_i}}\}$. For all $f \in \{c_1, c_2, \ldots, c_\alpha, r\}^{n_i}$, we set

$$A_i(f) := A_{j_1}(f) + A_{j_2}(f) - |\{v \in X_i \mid f(v) = r\}|.$$

Here G_i is simply the union of G_{j_1} and G_{j_2}. The cost is then the sum of the costs of G_{j_1} and G_{j_2}, except this counts those r-colored vertices twice that occur in both subgraphs. By property (3) of tree decompositions (Definition 1), these are exactly the r-colored vertices in X_i, so we can subtract their costs.

By property (1) of tree decompositions, for the root t of T it holds that $G_t = G$. Therefore, when reaching t, the minimum cost to color G corresponds to $\min\{A_t(f) \mid f \in \{c_1, c_2, \ldots, c_\alpha, r\}^{n_t}\}$. By property (2) of tree decompositions, this is in fact a valid coloring extension of L_S, since every edge where

the property of a coloring extension might be violated has been examined in some bag.

The computation of A_i for introduce and join nodes $i \in I$ needs $O((\alpha+1)^{n_i} \cdot (|X_i|+|E_i|))$ time, while, for forget nodes, it needs $O((\alpha+1)^{n_i+1} \cdot (|X_i|+|E_i|))$ time. Since $|n_i| \leq \omega+1$ for introduce and join nodes i and $|n_i| \leq \omega$ for forget nodes i and $\alpha \leq |S|$, we have the claimed running time. □

In Vertex (Weighted) Multiway Cut, we seek for a vertex subset whose removal leaves a graph where each connected component contains at most one terminal. Then, we have only one possible partition of S, namely, $|S|$ sets each containing exactly one terminal. Thus, the algorithm given in the proof of Theorem 7 solves Vertex (Weighted) Multiway Cut in the same time bound with $|C| = |S|$.

Corollary 1 Vertex Multiway Cut *on a graph $G = (V, E)$ with terminal set S and given tree decomposition of width ω can be solved in $O((|S|+1)^{\omega+1} \cdot \omega^2 \cdot |V|)$ time.*

Coming back to our algorithm for RVMC, the first phase of the algorithm enumerates all possible partitions of the terminal set S that separate all input terminal pairs. In the second phase, for each partition, the algorithm colors the terminal set according to the partition by using at most $|S|$ colors and, then, calls the dynamic programming algorithm for the Coloring Extension problem. The minimum of the outputs of the dynamic programming algorithm for all partitions is then the optimal solution for RVMC. By a simple traceback phase, one can easily construct the set of the vertices to be removed. The theorem then follows directly from the correctness and running times of the two phases. Directly multiplying the running times of both phases gives a running time of $O((|S|+1)^{|S|+\omega+1} \cdot \omega^2 \cdot |V|)$. However, we can save the "+1" for the following reason. We only need $|S|$ colors for the partition that has exactly one terminal in every set. For all other partitions, the dynamic programming uses at most $|S|-1$ colors. Hence, the overall running time of the algorithm amounts to $O((|S|+1)^{\omega+1} \cdot \omega^2 \cdot |V|) + (|S|^{|S|}-1) \cdot O(|S|^{\omega+1} \cdot \omega^2 \cdot |V|) = O(|S|^{|S|+\omega+1} \cdot \omega^2 \cdot |V|)$. In summary we arrive at the following main theorem.

Theorem 8 *Given an undirected graph $G = (V, E)$ with a tree decomposition of width ω,* Restricted Vertex Multicut *can be solved in $O(|S|^{|S|+\omega+1} \cdot \omega^2 \cdot |V|)$ time, where S is the terminal set.*

Note that the algorithms for Corollary 1 and Theorem 8 can be easily extended to handle weighted vertices within the same time bounds.

UVMC can be reduced to RVMC with the same number of terminals and the same treewidth (Section 1). Hence, the above algorithm also works for UVMC.

Corollary 2 *Given an undirected graph $G = (V, E)$ with a tree decomposition of width ω,* Unrestricted Vertex Multicut *can be solved in $O(|S|^{|S|+\omega+1} \cdot \omega^2 \cdot |V|)$ time, where S is the terminal set.*

In fact, the same approach can also be applied to EMC. Here, the optimization goal is to minimize the number of the "color-changing" edges, whose endpoints have different colors, while extending a coloring of the terminal set. The dynamic programming on the tree decomposition is almost the same. We omit the basically straightforward details.

Theorem 9 *Given an undirected graph $G = (V, E)$ with a tree decomposition of width ω,* Edge Multicut *can be solved in $O(|S|^{|S|+\omega+1} \cdot \omega^2 \cdot |V|)$ time, where S is the terminal set.*

Note that the fixed-parameter tractability of EMC (Theorem 9) was independently shown by Bentz [1].

5 Conclusions

We investigated the complexity of several Multicut problems and provided both hardness and tractability results. In further work relations between undirected and directed variants of Multicut were shown [24]: In particular, UVMC and RVMC can be reduced to any of directed UVMC, directed RVMC, and directed EMC by a reduction that preserves the parameter "number of edge/vertex deletions".

Many open questions remain. It is open for EMC, UVMC, and RVMC in general graphs whether they are fixed-parameter tractable with respect to the number of deletions k (see Table 1). Also, it would be interesting to examine complexity and algorithmic approaches for the vertex-deletion Multicut variants on other restricted graph classes such as bipartite graphs and planar graphs, as they were examined by Costa et al. [8] for EMC.

Acknowledgements

We thank three anonymous referees for valuable remarks that helped to significantly improve the presentation. In particular, we are grateful to a referee who pointed out a great simplification of the proof of Theorem 5.

Jiong Guo and Falk Hüffner were supported by the Deutsche Forschungsgemeinschaft, Emmy Noether research group PIAF (fixed-parameter algo-

rithms), NI 369/4.

References

[1] C. Bentz. Edge disjoint paths and multicut problems in graphs generalizing the trees. Technical Report 948, CNAM-Laboratoire Cédric, Paris, 2005. 3, 4, 17

[2] H. L. Bodlaender. Treewidth: Algorithmic techniques and results. In *Proc. 22nd MFCS*, volume 1295 of *Lecture Notes in Computer Science*, pages 19–36. Springer, 1997. 4, 5

[3] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209:1–45, 1998. 4, 5

[4] H. L. Bodlaender. Treewidth: Characterizations, applications, and computations. In *Proc. 32nd WG*, volume 4271 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006. 5, 12

[5] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976. 10, 11

[6] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: a Survey*. SIAM Monographs on Discrete Mathematics and Applications, 1999. 4, 10, 12

[7] R. D. Carr, S. Doddi, G. Konjevod, and M. V. Marathe. On the red-blue set cover problem. In *Proc. 11th SODA*, pages 345–353. ACM/SIAM, 2000. 12

[8] M. Costa, L. Létocart, and F. Roupin. Minimal multicut and maximal integer multiflow: a survey. *European Journal of Operational Research*, 162(1):55–69, 2005. 2, 17

[9] G. Călinescu, C. G. Fernandes, and B. Reed. Multicuts in unweighted graphs and digraphs with bounded degree and bounded tree-width. *Journal of Algorithms*, 48(2):333–359, 2003. 2, 3, 4, 5, 6, 7, 8, 9, 10

[10] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994. 2, 4, 12

[11] E. A. Dinic. Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Mathematics Doklady*, 11:1277–1280, 1970. 2

[12] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999. 5

[13] P. L. Erdős and L. A. Székely. Evolutionary trees: an integer multicommodity max-flow–min-cut theorem. *Advances in Applied Mathematics*, 13(4):375–389, 1992. 13

[14] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006. 5

[15] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979. 6

[16] N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997. 2, 4, 10

[17] N. Garg, V. Vazirani, and M. Yannakakis. Multiway cuts in node weighted graphs. *Journal of Algorithms*, 50(1):49–61, 2004. 3

[18] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980. 10

[19] J. Guo, F. Hüffner, E. Kenar, R. Niedermeier, and J. Uhlmann. Complexity and exact algorithms for Multicut. In *Proc. 32nd SOFSEM*, volume 3831 of *Lecture Notes in Computer Science*, pages 137–147. Springer, 2006. 11, 12

[20] J. Guo and R. Niedermeier. Fixed-parameter tractability and data reduction for Multicut in Trees. *Networks*, 46(3):124–135, 2005. 2, 4, 7

[21] J. Guo and R. Niedermeier. Exact algorithms and applications for Tree-Like Weighted Set Cover. *Journal of Discrete Algorithms*, 4(4):608–622, 2006. 2

[22] T. C. Hu. Multicommodity networks flow. *Operations Research*, 9:898–900, 1963. 2

[23] A. Itai. Two-commodity flow. *Journal of the ACM*, 25:596–611, 1978. 2

[24] E. Kenar and J. Uhlmann. Multicut in graphs. Student research project, WSI für Informatik, Universität Tübingen, Germany, June 2005. 17

[25] T. Kloks. *Treewidth: Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994. 5

[26] D. Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394–406, 2006. 2, 3, 4

[27] T. A. McKee and F. R. McMorris. *Topics in Intersection Graph Theory*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, 1999. 3

[28] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988. 12

[29] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. 5

[30] B. A. Reed. Algorithmic aspects of tree width. In B. A. Reed and C. L. Sales, editors, *Recent Advances in Algorithms and Combinatorics*, pages 85–107. Springer, 2003. 4, 5

[31] N. Robertson and P. D. Seymour. Graph minors. II: Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986. 4

[32] J. A. Telle and A. Proskurowski. Practical algorithms on partial k-trees with an application to domination-like problems. In *Proc. 3rd WADS*, volume 709 of *Lecture Notes in Computer Science*, pages 610–621. Springer, 1993. 4