Proc. 32nd SOFSEM-06, Vol. 3831 in LNCS, pp. 303-312, Springer, 2006

Complexity and Exact Algorithms for Multicut*

Jiong Guo[1], Falk Hüffner[1], Erhan Kenar[2], Rolf Niedermeier[1], and Johannes Uhlmann[2]

[1] Institut für Informatik, Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany. {guo,hueffner,niedermr}@minet.uni-jena.de
[2] Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Sand 13, D-72076 Tübingen, Germany. {kenar,johannes}@informatik.uni-tuebingen.de

Abstract. The Multicut problem 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.

1 Introduction

Motivation and previous results. Multicut in graphs is a fundamental network design problem. It models questions concerning the reliability and 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 [2] for a recent survey.

The major part of the literature deals with the "edge deletion variant" of Multicut (Edge Multicut) [2, 6–8] whereas 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 [3, 9]. Călinescu, Fernandes, and Reed [3] 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.

* Research supported by the Deutsche Forschungsgemeinschaft (DFG), Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4.

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.

Călinescu et al. show 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 give a polynomial-time approximation scheme (PTAS) for UVMC in bounded treewidth graphs. Marx [9] extends the results for UVMC (which he calls MINIMUM TERMINAL PAIR SEPARATION) by providing an $O(2^{k\ell} \cdot k^k \cdot 4^{k^3} \cdot |G|^{O(1)})$ 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).

Our results. We continue and complement the work of Călinescu et al. [3] and Marx [9] 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 $O(2^k \cdot |G| \cdot \ell)$ (again, ℓ is the number of terminal pairs). Whereas in trees UVMC is polynomial-time solvable but RVMC is NP-complete [3], we have the surprising result that UVMC is NP-complete in interval graphs but RVMC is polynomial-time solvable here.[3] 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 $O(|S|^{|S|+\omega+1} \cdot |G|)$ time on graphs of treewidth ω, where S denotes the set of terminal vertices; 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 the parameter $|S|$ or ω. This fixed-parameter tractability result directly transfers to UVMC as well; indeed, it also works for the EDGE MULTICUT (EMC) variant. Finally, for EDGE MULTICUT we also prove NP-completeness in caterpillar graphs with maximum vertex degree five.

Table 1 summarizes most of the presented results.

[3] More specifically, the NP-completeness result for UVMC even holds in interval graphs of pathwidth four.

Table 1. 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 [6]	NP-c (Thm. 4)	P (Thm. 5)
Trees		NP-c [6]	P (Sect. 2)	NP-c [3]
	k	FPT [8]	—	FPT (Thm. 2)
General graphs		NP-c [4]	NP-c [3]	NP-c [3]
	k	open	open	open
	$\|S\|$	NP-c [4]	FPT [9]	NP-c (Thm. 6)
	ω	NP-c [6]	NP-c [3]	NP-c [3]
	ω & $\|S\|$	FPT (Thm. 9)	FPT (Cor. 1)	FPT (Thm. 8)

Preliminaries. We introduce some additional terminology. By default, we consider only undirected graphs $G = (V, E)$ without self-loops. A graph is an *interval graph* if we can label its vertices by intervals of the real line such that there is an edge between two vertices iff their intervals intersect. A tree is called *caterpillar* if all vertices with degree at least three have at most two neighbors of degree two or greater. For any graph $G = (V, E)$, we can construct its *line graph* as $(E, \{\{e_1, e_2\} \in E \mid e_1 \cap e_2 \neq \emptyset\})$. We use $G[V']$ to denote the subgraph of G induced by the vertices $V' \subseteq V$. A set of vertices $V' \subseteq V$ is called *vertex separator* if $G[V \setminus V']$ has more connected components than G.

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: $\bigcup_{i \in I} X_i = V$; for every edge $\{u, v\} \in E$, there is an $i \in I$ such that $\{u, v\} \subseteq X_i$; and 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.

A problem of size n is called *fixed-parameter tractable* (FPT) with respect to a parameter k if it can be solved in $f(k) \cdot n^{O(1)}$ time, where f is a function solely depending on the parameter k.

Due to the lack of space, some proofs had to be omitted.

2 Trees

Unrestricted Vertex Multicut in trees is trivially solvable in $O(|V| \cdot |H|)$ time: Root the tree at an arbitrary vertex. Then, compute the least common ancestors for all terminal pairs in H and sort these ancestors in a list L according to the decreasing order of their depth in the rooted tree. Finally, while $L \neq \emptyset$, remove the first element of L and its corresponding vertex from T and delete all

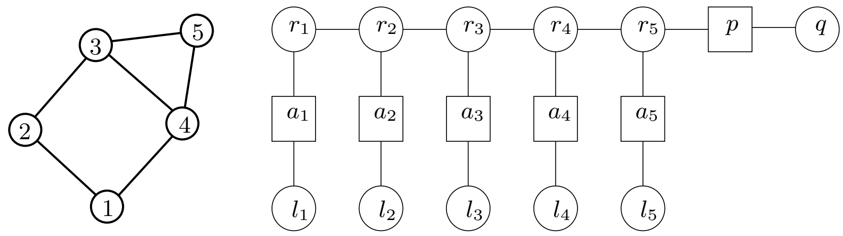

Fig. 1. An example for the reduction from VERTEX COVER to RVMC. The left figure is a VERTEX COVER instance and the right is the corresponding RVMC instance with $H = \{(l_1, l_2), (l_2, l_3), (l_3, l_4), (l_4, l_1), (l_3, l_5), (l_4, l_5)\} \cup \{(r_i, q) \mid 1 \le i \le 5\}$. Only the rectangular vertices can be deleted (all others are terminals).

separated terminal pairs from H and their least common ancestors from L. The solution is then the removed vertices. We omit further details.

Călinescu et al. [3] show that RVMC is NP-complete in trees with maximum vertex degree four by giving a reduction from EMC in binary trees. It is easy to observe that RVMC on trees with maximum vertex degree two, i.e., paths, can be solved in polynomial time. The complexity of RVMC in trees with maximum vertex degree three remained open. Here we close this gap.

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, which for a graph $G = (V = \{v_1, v_2, \ldots, v_n\}, E)$ and $k \ge 0$ asks whether there is a set of vertices $V' \subseteq V$ with $|V'| \le k$ such that for every edge $\{v, w\} \in E$ at least one of v and w is in V'. Construct the tree $T = (W, F)$ with

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

and

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

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. $\square$

In the following, we show that RVMC in trees can be solved in $O(2^k \cdot |V| \cdot |H|)$ time, where k denotes the allowed number of vertex removals. The basic idea is to modify the above polynomial-time algorithm for UVMC in trees into a depth-bounded search tree algorithm.

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.

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. 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.

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 $O(2^k)$.

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

3 Interval Graphs

As mentioned in the introduction, 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 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. However, the class of interval graphs is an exception, that is, UVMC is NP-complete in interval graphs while RVMC is solvable in polynomial time:

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 caterpillars and then reduce EMC in caterpillars to UVMC in interval graphs. We use a reduction from 3-SAT, which is similar to the reduction used to show the NP-completeness of EMC in binary trees [3, Theorem 6.1].

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

The second reduction from EMC in caterpillars with maximum vertex degree five to UVMC in interval graphs with pathwidth four is—analogous to [3]—executed by constructing the line graph of the caterpillar:

Theorem 4. UNRESTRICTED VERTEX MULTICUT *in interval graphs with pathwidth at least four is NP-complete.*

Restricted Vertex Multicut in Interval Graphs. In contrast to EMC[4] and UVMC, which are NP-complete for interval graphs even with bounded pathwidth, we now give a dynamic programming algorithm solving RVMC in interval graphs in polynomial time.

For an interval graph $G = (V, E)$, we can construct a path decomposition in $O(|V| + |E|)$ time such that each bag one-to-one corresponds to a maximal clique of G (see Booth and Lueker [1]). The minimal vertex separators of G are the intersections of two neighboring bags in the path decomposition and are also cliques. The following lemma shows that the minimal vertex separators are crucial for solving RVMC in interval graphs.

Lemma 1. *Any optimal solution of RVMC in interval graphs consists of a selection of the minimal vertex separators, that is, for each vertex v in an optimal solution C for RVMC in an interval graph G, there is a minimal vertex separator Y of G such that $v \in Y$ and $Y \subseteq C$.*

Based on Lemma 1, we only consider the minimal vertex separators of G. Note that we exclude the vertex separators containing terminals since such vertex separators cannot be contained in an optimal solution for RVMC. Let $Y_1, Y_2, \ldots, Y_r$ with $r \leq |V|$ be the minimal vertex separators obtained by the path decomposition of G. For each $1 \leq i \leq r$ we define $P_i \subseteq H$ as the set containing the terminal pairs that are disconnected in $G[V \setminus Y_i]$. Let $H_i := \bigcup_{1 \leq j \leq i} P_j$. Note that $H_r = H$; otherwise, the given instance has no solution. Due to the third property of the path decomposition (see Sect. 1), the minimal separators can be linearly ordered such that they fulfill the following "*consistency condition*": $Y_i \cap Y_l \subseteq Y_j$ for all Y_i, Y_j, Y_l with $i \leq j \leq l$. In addition, the sets P_i associated with Y_i, $1 \leq i \leq r$, fulfill also this consistency condition.

Now, we can formulate RVMC as a covering problem: Solving RVMC in interval graphs is equivalent to finding a subset Z of $\{1, 2, \ldots, r\}$ with $\bigcup_{i \in Z} P_i = H$ and $|\bigcup_{i \in Z} Y_i| \leq k$. Observe that, although at first sight they are very similar, there is a decisive difference to the classical SET COVER problem.[5] In our case, we have two subset systems, one is the set H and its subsets $P_1, P_2, \ldots, P_r$ and the other is V and its subsets $Y_1, Y_2, \ldots, Y_r$. Each P_i is associated with the subset of V with the same index, i.e., Y_i. The task is to select some sets from $\mathcal{P} := \{P_1, P_2, \ldots, P_r\}$ to cover H. However, instead of minimizing the number of selected subsets from $\mathcal{P}$ as in SET COVER, we minimize $|\bigcup_{P_i \in \mathcal{P}'} Y_i|$ where $\mathcal{P}'$ denotes the set of the selected subsets from $\mathcal{P}$.

[4] EMC is NP-complete even in stars [6], which are interval graphs with pathwidth one.

[5] The in general NP-complete SET COVER problem can be solved in polynomial time on instances where the subsets in the subset collection can be linearly ordered such that they fulfill the consistency condition described above [10].

Observe that the minimal separators are not pairwise disjoint, i.e., there may be two Y_i and Y_j with $i \neq j$ and $Y_i \cap Y_j \neq \emptyset$. On the one hand, this forbids assigning to each P_i a weight equal to $|Y_i|$ and solving a "weighted" version of Set Cover, since $|\bigcup_{P_i \in \mathcal{P}'} Y_i|$ is not always equal to $\sum_{P_i \in \mathcal{P}'} |Y_i|$. On the other hand, for two separators Y_i and Y_j with $i \neq j$ and $Y_i \cap Y_j \neq \emptyset$, the selection of Y_j may affect the decision concerning the selection of Y_i and vice versa.

The key idea of the algorithm for this special covering problem is to exploit the consistency property of the minimal separators Y_i and their associated sets P_i: Order the minimal vertex separators Y_i and the associated sets P_i linearly such that they fulfill the consistency property. Then, the algorithm processes this linear ordering from $i = 1$ to $i = r$. At each i, it computes the best "local solution" to cover H_i by using only $P_1, \ldots, P_i$. As mentioned above, the selection of Y_j with $j > i$ may affect the decision concerning the selection of Y_i. To cope with this, the algorithm computes not only one local solution but $r - i + 1$ values for each i. The first value B_i represents the best local solution under the assumption that no j with $i < j \leq r$ and $Y_j \cap Y_i \neq \emptyset$ will be added into the global solution. Each of the other $r - i$ values $F_{i,j}$ represents, for each $i < j \leq r$, the best local solution under the assumption that j but no l with $i < l < j$ and $Y_l \cap Y_i \neq \emptyset$ will be added to the global solution. Note that due to the consistency condition, $F_{i,j}$ is equal to the best local solution under the assumption that j and l will be added to the global solution for any $l > j$.

When reaching r, there are at most two cases to consider: r is in the global solution or it is not. If it is not, then the local solution B_{r-1} turns out to be the global solution; otherwise, $|Y_r| + \min_{1 \leq i < r}\{F_{i,r} \mid (H \setminus P_r) \subseteq H_i\}$ is the global solution.

In the following, we give the formal description of the algorithm.

For each i, $1 \leq i \leq r$, we will compute B_i and $F_{i,j}$, $i < j \leq r$, such that the following invariants hold:

$$B_i = \min\{\, |\bigcup_{l \in Z} Y_l| \mid Z \subseteq \{1, \ldots, i\} \text{ and } H_i = \bigcup_{l \in Z} P_l \,\},$$
$$F_{i,j} = \min\{\, |\bigcup_{l \in Z} (Y_l \setminus Y_j)| \mid Z \subseteq \{1, \ldots, i\} \text{ and } H_i = \bigcup_{l \in Z} P_l \,\}.$$

In order to simplify the presentation, we introduce $Y_0 := \emptyset$, $P_0 := \emptyset$, $H_0 := \emptyset$, $B_0 := 0$, and $F_{0,j} := 0$ for all $1 \leq j \leq r$. We start with the initialization $B_1 := |Y_1|$ and for all $1 < j \leq r$ let $F_{1,j} := |Y_1 \setminus Y_j|$.

When reaching i with $1 < i < r$, we consider two cases.

Case 1. $P_i \not\subseteq P_{i-1}$.

We set

$$B_i := |Y_i| + \min_{0 \leq l < i}\{\, F_{l,i} \mid (H_i \setminus P_i) \subseteq H_l \,\};$$
$$F_{i,j} := |Y_i \setminus Y_j| + \min_{0 \leq l < i}\{\, F_{l,i} \mid (H_i \setminus P_i) \subseteq H_l \,\}, \text{ for each } i < j \leq r.$$

This computation is correct since we have to take i to have a local solution. Then, B_i is set equal to the sum of $|Y_i|$ and the minimum of the local solutions

to cover $H_i \setminus P_i$ under the assumption that i is already a part of the solution. The value of $F_{i,j}$ is set analogously.
Case 2. $P_i \subseteq P_{i-1}$.
We set

$$B_i := \min\{\, B_{i-1}, |Y_i| + \min_{0\le l<i}\{\, F_{l,i} \mid (H_i \setminus P_i) \subseteq H_l \,\}\};$$
$$F_{i,j} := \min\{\, F_{i-1,j}, |Y_i \setminus Y_j| + \min_{0\le l<i}\{\, F_{l,i} \mid (H_i \setminus P_i) \subseteq H_l \,\}\}, \text{ for each } i < j \le r.$$

In this case, we choose the minimum of the two alternatives of adding i to the solution or not. Therefore, the correctness follows from the correctness of Case 1.

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

4 General Graphs and Bounded Treewidth

In this section, we present a fixed-parameter algorithm for RVMC in general graphs with treewidth and the number of terminals as parameters. Marx [9] shows that UVMC is fixed-parameter tractable with respect to the number of vertex removals and the number of terminal pairs.

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. For the 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 more than two input terminal pairs [4].

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

Now we know that there is no hope for a fixed-parameter algorithm for RVMC with respect to the single parameter treewidth or number of terminals. In the following, we present the fixed-parameter algorithm for RVMC with treewidth *and* the number of the terminals as parameters. The basic idea of this algorithm comes from the observation 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 observation, 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 $O(|S|^{|S|})$ many partitions of the terminal set S. To check whether a partition separates the given terminal pairs in H can be done in $O(|H|)$ time. Then, the run 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:[6]

COLORING EXTENSION
Input: An undirected graph $G = (V, E)$, a set of terminals $S \subseteq V$, and a pre-coloring $L_S : S \to C$ with the colors from a set C where $|C| \leq |S|$.
Task: Find an extension $L_{G,S}$ of L_S with the colors from $C \cup \{r\}$ where $r \notin C$ such that

1. for every $s \in S$, $L_{G,S}(s) = L_S(s)$,
2. for every edge $\{u, v\} \in E$, either $L_{G,S}(u) = L_{G,S}(v)$ or $L_{G,S}(u) = r$ or $L_{G,S}(v) = r$, and
3. the cost $|\{v \in V \mid L_{G,S}(v) = r\}|$ is minimized.

Assume we have a fixed partition of the terminal set. If we assign to every terminal in a set of this partition a color from C, then a solution of the coloring problem ensures that every path between two terminals with different initial colors has to pass through at least one vertex v with $L_{G,S}(v) = r$. This implies that the removal of the vertices with color r separates the sets, which is a solution for the RVMC problem.

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

In summary, the first phase of the algorithm for RVMC 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 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 main theorem then follows directly from the correctness and run times of the two phases.

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

UVMC can be reduced to RVMC with the same number of terminals and the same treewidth (Sect. 3). Therefore, the above algorithm also works for UVMC.

[6] A similar coloring problem is defined by Erdős and Székely [5]. Note that here we have a different cost function.

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

Actually, the same approach can also be applied to EMC. Here, the 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 details.

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

References

1. 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*, 13:335–379, 1976.
2. 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.
3. G. Călinescu, C. G. Fernandes, and B. Reed. Multicuts in unweighted graphs and digraphs with bounded degree and bounded tree-width. *Journal of Algorithms*, 48:333–359, 2003.
4. 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.
5. P. L. Erdős and L. A. Székely. Evolutionary trees: an integer multicommodity max-flow–min-cut theorem. *Advances in Applied Mathematics*, 13:375–389, 1992.
6. N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.
7. J. Guo and R. Niedermeier. Exact algorithms and applications for Tree-Like Weighted Set Cover. To appear in *Journal of Discrete Algorithms*, 2005.
8. J. Guo and R. Niedermeier. Fixed-parameter tractability and data reduction for Multicut in Trees. *Networks*, 46(3):124–135, 2005.
9. D. Marx. Parameterized graph separation problems. In *Proc. 1st IWPEC*, volume 3162 of *LNCS*, pages 71–82. Springer, 2004. Long version to appear in *Theoretical Computer Science*.
10. A. F. Veinott and H. M. Wagner. Optimal capacity scheduling. *Operations Research*, 10:518–532, 1962.