



TECHNICAL REPORT



INSTITUT FÜR INFORMATIONSSYSTEME
ABTEILUNG DATENBANKEN UND ARTIFICIAL INTELLIGENCE

Multicut Algorithms via Tree Decompositions

DBAI-TR-2010-67

Reinhard Pichler Stefan Rümmele Stefan Woltran

Institut für Informationssysteme
Abteilung Datenbanken und
Artificial Intelligence
Technische Universität Wien
Favoritenstr. 9
A-1040 Vienna, Austria
Tel: +43-1-58801-18403
Fax: +43-1-58801-18492
sekret@dbai.tuwien.ac.at
www.dbai.tuwien.ac.at

DBAI TECHNICAL REPORT
2010



Multicut Algorithms via Tree Decompositions

Reinhard Pichler¹ Stefan Rümmele² Stefan Woltran³

Abstract. Various forms of multicut problems are of great importance in the area of network design. In general, these problems are intractable. However, several parameters have been identified which lead to fixed-parameter tractability (FPT). Recently, Gottlob and Lee have proposed the treewidth of the structure representing the graph and the set of pairs of terminal vertices as one such parameter. In this work, we show how this theoretical FPT result can be turned into efficient algorithms for optimization, counting, and enumeration problems in this area.

¹Institute for Information Systems 184/2, Technische Universität Wien, Favoritenstrasse 9-11, 1040 Vienna, Austria. E-mail: pichler@dbai.tuwien.ac.at

²Institute for Information Systems 184/2, Technische Universität Wien, Favoritenstrasse 9-11, 1040 Vienna, Austria. E-mail: ruemmele@dbai.tuwien.ac.at

³Institute for Information Systems 184/2, Technische Universität Wien, Favoritenstrasse 9-11, 1040 Vienna, Austria. E-mail: woltran@dbai.tuwien.ac.at

Acknowledgements: The authors acknowledge support by the Austrian Science Fund (FWF), project P20704-N18, and the Vienna Science and Technology Fund (WWTF), project ICT08-028.

This is an extended version of a paper published in the Proceedings of the 7th International Conference on Algorithms and Complexity (CIAC 2010).

Copyright © 2010 by the authors

1 Introduction

Multicut problems [1] are of great importance in the area of network design – with applications to telecommunications, routing, VLSI design and transportation. An instance of a multicut problem is given by an undirected graph $G = (V, E)$ and a set H of pairs of *terminal* vertices. A *solution* of the EDGE MULTICUT (EMC) problem is a set of edges whose removal disconnects each terminal pair. In case of the RESTRICTED (resp. UNRESTRICTED) VERTEX MULTICUT problem, the *solutions* are sets of non-terminal vertices (resp. of arbitrary vertices) whose removal disconnects each terminal pair. Each notion of solutions naturally gives rise to an optimization problem (“find the cardinality of a minimal solution”). Equipping, in addition, edges (resp. vertices) with some positive integer naturally leads to weighted versions of this optimization problem (“find the total weight of a minimal solution”). All these variants of multicut problems are intractable (the corresponding decision problems whether a solution with a given cardinality resp. total weight exists are NP-complete [2, 3]), even for graphs of bounded treewidth [2, 4]. For planar graphs with outer terminals and fixed cardinality $|H|$, EMC is solvable in polynomial-time [5].

An important approach in dealing with intractable problems is to search for *fixed-parameter tractability* (FPT), see e.g. [6]. Thereby one tries to confine the combinatorial explosion to certain problem *parameters*. More formally, we speak of FPT w.r.t. a parameter k , if a problem is solvable in time $f(k) \cdot n^{O(1)}$, where n denotes the size of the input instance. The function f is usually exponential but only depends on k . In case of multicut problems, various such parameters have been studied like solution size [7, 8], cardinality $|H|$ plus solution size [9, 10], $|H|$ plus the treewidth of the graph G [11, 12], or the treewidth of the structure representing both G and H [13]. The result in [13] was proved by showing that the solutions to any of the above multicut problems can be described by a monadic second-order (MSO) formula over the structure representing G and H . The FPT follows by an extension of Courcelle’s Theorem [14] proved in [15]: optimization problems expressible by an MSO formula over the input structures are fixed-parameter tractable w.r.t. the treewidth of these structures. The proof of this result in [15] is constructive – yielding algorithms whose running time is non-elementary in terms of the number of quantifier alternations of the MSO formula. For the MSO formula in [13], we would thus end up with time and space requirements which are at least double exponential in the treewidth. The goal of our paper is to construct more efficient algorithms in case of bounded treewidth of the structure representing G and H . Our main results are as follows (Due to space constraints, we omit proofs in this paper; full proofs are given in [16]):

- *Optimization Problem.* We present a new dynamic-programming based algorithm for computing the optimal value of the EDGE MULTICUT problem which works by partitioning the vertices into disconnected parts. The resulting algorithm runs in time $O(2^{2w^* \log w^*} \cdot \|(G, H)\|)$, where w^* denotes the treewidth of the input structure and $\|(G, H)\|$ denotes its size. Therefore this algorithm is in fixed-parameter linear time and single exponential in terms of the treewidth.
- *Counting and Enumeration.* We come up with a refinement of the optimization algorithm in order to *count* the number of optimal solutions and also to

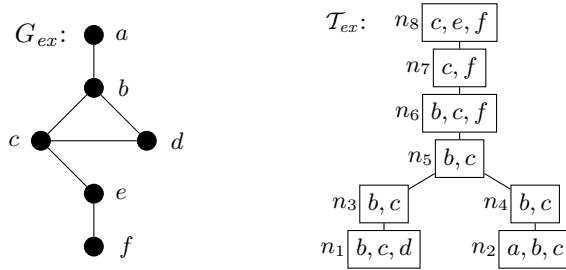


Fig. 1. Example graph G_{ex} and normalized tree decomposition \mathcal{T}_{ex} of G_{ex} .

enumerate all optimal solutions. Determining the number of optimal solutions is possible in time $O(2^{2((w^*)^2 + w^* \log w^*)} \cdot \|(G, H)\|)$. Moreover, the enumeration of all optimal solutions is feasible with a delay (i.e., the time needed to compute the first respectively the next solution) that also fits into this bound.

2 Preliminaries

We briefly recall some basic definitions on treewidth and multicut problems.

Tree Decomposition and Treewidth. A *tree decomposition* of a graph $G = (V, E)$ is a pair (T, χ) , where T is a tree and χ maps each node n of T (we use $n \in T$ as a shorthand below) to a *bag* $\chi(n) \subseteq V$ with the following properties: (1) for each $v \in V$, there is an $n \in T$, s.t. $v \in \chi(n)$; (2) for each $\{v, w\} \in E$, there is an $n \in T$, s.t. $v, w \in \chi(n)$; (3) for each $n_1, n_2, n_3 \in T$, s.t. n_2 lies on the path from n_1 to n_3 , $\chi(n_1) \cap \chi(n_3) \subseteq \chi(n_2)$ holds.

A rooted tree decomposition (T, χ) is called *normalized* (or *nice*) [17], if (1) each $n \in T$ has at most two children; (2) for each $n \in T$ with two children n_1, n_2 , $\chi(n) = \chi(n_1) = \chi(n_2)$; and (3) for each $n \in T$ with one child n' , $\chi(n)$ and $\chi(n')$ differ in exactly one element; in other words the symmetric set-difference of $\chi(n)$ and $\chi(n')$ is a singleton $\{v\}$ with $v \in V$.

The *width* of a tree decomposition is defined as the cardinality of its largest bag minus one. The *treewidth* of a graph G , denoted as $tw(G)$, is the minimum width over all tree decompositions of G . For arbitrary but fixed $w \geq 1$, it is feasible in linear time (with huge constants though) to decide if a graph has treewidth $\leq w$ and, if so, to compute a tree decomposition of width w [18]. Moreover, there exist sophisticated heuristics to efficiently construct tree decompositions of small width [19, 20]. W.l.o.g., we may assume that every tree decomposition is normalized, since this normalization can be obtained in linear time without increasing the width [17]. We thus distinguish between four types of nodes: (vertex) introduction (I), (vertex) removal (R), branch (B), and leaf (L) nodes. The first two types will usually be augmented with the vertex v which is removed or added compared to the bag of the child node, i.e., (vR) and (vI) .

Example 1. Figure 1 shows an example graph G_{ex} together with a normalized tree decomposition \mathcal{T}_{ex} of width 2. Note that the treewidth of G_{ex} is indeed 2,

since only trees (or forests) have treewidth = 1. Examples for node types are n_1 as (L) node, n_3 as (dR) node, n_5 as (B) node, and n_6 , as (fI) node.

Multicut Problems. Since we are dealing with undirected graphs $G = (V, E)$, we consider terminal pairs $h \in H$ as sets $h \subseteq V$ of cardinality two. In [13], the treewidth w^* of a structure representing both the graph G and the set H of pairs of terminals was introduced as a parameter of the multicut problems. Alternatively, w^* can also be defined as $w^* = tw(G')$ with $G' = (V, E \cup H)$.

In this paper, we focus on the edge multicut problem (EMC) without weights. Given an instance (G, H) of the EMC problem with $G = (V, E)$, we define a cut for (G, H) as a set $F \subseteq E$ of edges, such that for each $\{h_1, h_2\} \in H$, h_1 and h_2 are disconnected in $(V, E \setminus F)$. $Cuts(G, H)$ will denote the set of cuts for (G, H) and $MinCuts(G, H)$ is the set of minimum cuts for (G, H) , i.e. $MinCuts(G, H) = \{F \in Cuts(G, H) : \forall F' \in Cuts(G, H) : |F| \leq |F'|\}$.

Example 2. Consider the instance (G_{ex}, H_{ex}) of the EMC problem, where $G_{ex} = (V, E)$ is the graph in Figure 1 and $H_{ex} = \{\{a, b\}, \{c, d\}\}$. As is easily verified, the sets $\{\{a, b\}, \{b, c\}, \{c, d\}\}$ and $\{\{a, b\}, \{b, d\}, \{c, d\}\}$ are the minimum cuts for (G_{ex}, H_{ex}) . In this simple example, we have $(V, E \cup H_{ex}) = (V, E)$. Hence, the treewidth w^* of (G_{ex}, H_{ex}) is equal to the treewidth of G_{ex} . We thus can use the tree decomposition of G_{ex} in Figure 1 also as a tree decomposition of (G_{ex}, H_{ex}) .

For the remainder of the paper, we fix some notation: $G = (V, E)$ and H always refer to the currently considered instance (G, H) of a multicut problem and we denote the size of a reasonable representation by $\|(G, H)\|$. $\mathcal{T} = (T, \chi)$ denotes a normalized tree decomposition of (G, H) and n_{root} refers to the root node of T . For any node $n \in T$, we denote by T_n the subtree of T rooted at n and we use $\chi(T_n) := \bigcup_{m \in T_n} \chi(m)$. Hence, we always have $V = \chi(T_{n_{root}})$.

3 Dynamic programming for edge multicut

In this section, we give a dynamic programming algorithm for the EMC problem to compute the number of edges a minimum cut requires. As well, we will sketch at the end of the section, how this algorithm can be used to compute one solution of a given EMC problem. However, we will also observe that the algorithm is not powerful enough to count or enumerate (without repetitions) all solutions of the given EMC problem. We will overcome this problem later in Section 4.

Our central objects in this section are so-called *colorings* (i.e., a special form of partitions), which partition subsets of V into several sets (colors), such that no terminal pair $(h_1, h_2) \in H$ appears in the same color. The intended meaning of a coloring is thus a separation of the vertices into components. Indeed, such a separation induces a cut in a natural way. For our considerations in Section 4, note that vertices of the same color are not necessarily connected in such a graph.

Definition 1. A coloring over a set $W \subseteq V$ of vertices is a partition $\mathcal{C} = \{C_1, \dots, C_k\}$ of W , such that $u, v \in C_i$ implies $\{u, v\} \notin H$. Moreover, we denote the set of all colorings over W by $\mathbf{C}(W)$. Finally, we define $\Gamma(\mathcal{C}) = E \cap \{\{u, v\} : u \in C_i, v \in C_j, i \neq j\}$ as the cut induced by \mathcal{C} .

Proposition 1. $\text{Cuts}(G, H) = \bigcup_{\mathcal{C} \in \mathbf{C}(V)} \{F : \Gamma(\mathcal{C}) \subseteq F \subseteq E\}$.

For our purposes, colorings related to the bags of the tree decomposition are of particular interest. For a coloring \mathcal{C} over $\chi(n)$, it is natural to extend \mathcal{C} to a coloring over $\chi(T_n)$. We will thus later be able to extend colorings over $\chi(n_{root})$ to colorings over V . The necessary concepts are introduced below.

Definition 2. Given sets $W \subseteq W' \subseteq V$, $\mathcal{C} \in \mathbf{C}(W)$, and $\mathcal{C}' \in \mathbf{C}(W')$. We say that \mathcal{C}' extends \mathcal{C} , in symbols $\mathcal{C} \leq \mathcal{C}'$, in case $\mathcal{C} = \{\mathcal{C}' \cap W : \mathcal{C}' \in \mathcal{C}'\} \setminus \{\emptyset\}$.

Definition 3. Given a node $n \in T$, we call a coloring over $\chi(n)$ also a coloring of n and we write $\mathbf{C}(n)$ instead of $\mathbf{C}(\chi(n))$. For $\mathcal{C} \in \mathbf{C}(n)$, we define

$$\begin{aligned} \text{ext}_n(\mathcal{C}) &= \{\mathcal{C}' \in \mathbf{C}(\chi(T_n)) : \mathcal{C} \leq \mathcal{C}'\} \\ \sigma_n(\mathcal{C}) &= \min\{|\Gamma(\mathcal{C}')| : \mathcal{C}' \in \text{ext}_n(\mathcal{C})\} \\ \mathcal{E}_n(\mathcal{C}) &= \{\Gamma(\mathcal{C}') : \mathcal{C}' \in \text{ext}_n(\mathcal{C}) \text{ s.t. } |\Gamma(\mathcal{C}')| = \sigma_n(\mathcal{C})\}. \end{aligned}$$

Intuitively, $\text{ext}_n(\mathcal{C})$ yields all colorings over $\chi(T_n)$ which extend the coloring \mathcal{C} . Note that such an extension always exists: Indeed, for nodes $\{u, v\} \in H$ contained in $\chi(n)$ this is assured by definition of colorings. Moreover, for nodes $\{u, v\} \in H$ which appear below n , we just have to assign different colors to u and v . Furthermore, $\sigma_n(\mathcal{C})$ gives the cardinality of the smallest induced cut among these extended colorings; and $\mathcal{E}_n(\mathcal{C})$ actually yields all these minimum cuts.

Example 3. The first three columns of the table in Figure 2 show the colorings $\mathcal{C} = \{C_1, C_2, \dots\}$ for all nodes n of our example tree decomposition, together with the value $\sigma_n(\mathcal{C})$. Due to space limitations, we will reuse this table in Section 4. Until then, the other columns can be ignored. Note that for the root node, n_8 , coloring $\mathcal{C} = \{c, e, f\}$ with minimal value $\sigma_{n_8}(\mathcal{C}) = 3$ yields the minimum cuts of our example. In fact, $\mathcal{E}_{n_8}(\mathcal{C}) = \{\{a, b\}, \{b, c\}, \{c, d\}\}, \{\{a, b\}, \{b, d\}, \{c, d\}\}$.

The last observation in the above example leads us to the following lemma.

Lemma 1. Let \mathbf{C}_{min} be the set of all $\mathcal{C} \in \mathbf{C}(n_{root})$ such that no other coloring \mathcal{C}' of n_{root} satisfies $\sigma_{n_{root}}(\mathcal{C}') < \sigma_{n_{root}}(\mathcal{C})$. Then, $\text{MinCuts}(G, H) = \bigcup_{\mathcal{C} \in \mathbf{C}_{min}} \mathcal{E}_{n_{root}}(\mathcal{C})$.

In other words, to compute the size of a minimum cut, it is sufficient to derive the values $\sigma_{n_{root}}(\mathcal{C})$ of all colorings $\mathcal{C} \in \mathbf{C}(n_{root})$. However, we do not want to compute $\sigma_{n_{root}}(\mathcal{C})$ directly with the help of Definition 3, but establish $\sigma_n(\mathcal{C})$ for all nodes $n \in T$ in a bottom-up fashion. We use the following concept.

Definition 4. For $\mathcal{C} \in \mathbf{C}(n)$ with $n \in T$, define $\pi_n(\mathcal{C})$ recursively as follows:

- If n is an (L) node, $\pi_n(\mathcal{C}) = |\Gamma(\mathcal{C})|$;
- If n is a (vR) node with child n' , $\pi_n(\mathcal{C}) = \min\{\pi_{n'}(\mathcal{C}') : \mathcal{C}' \in \mathbf{C}(n'), \mathcal{C} \leq \mathcal{C}'\}$;
- If n is a (vI) node with child n' , $\pi_n(\mathcal{C}) = \pi_{n'}(\mathcal{C}') + |\{u : \{u, v\} \in \Gamma(\mathcal{C})\}|$, where $\mathcal{C}' \in \mathbf{C}(n')$ is the unique coloring with $\mathcal{C}' \leq \mathcal{C}$;
- If n is a (B) node with children n', n'' , $\pi_n(\mathcal{C}) = \pi_{n'}(\mathcal{C}) + \pi_{n''}(\mathcal{C}) - |\Gamma(\mathcal{C})|$.

Lemma 2. For any node $n \in T$ and any $\mathcal{C} \in \mathbf{C}(n)$, $\sigma_n(\mathcal{C}) = \pi_n(\mathcal{C})$.

	C_1	C_2	C_3	σ	$\Delta_1 : \sigma : \#$	$\Delta_2 : \sigma : \#$
n_1	$\{b, c\}$	$\{d\}$		2	$\{\{C_1, C_2\}\} : 2 : 1$	
	$\{b, d\}$	$\{c\}$		2	$\{\{C_1, C_2\}\} : 2 : 1$	
	$\{b\}$	$\{c\}$	$\{d\}$	3	$\{\{C_2, C_3\}\} : 3 : 1$	
n_2	$\{a\}$	$\{b, c\}$		1	$\{\{C_1, C_2\}\} : 1 : 1$	
	$\{a, c\}$	$\{b\}$		2		
	$\{a\}$	$\{b\}$	$\{c\}$	2	$\{\{C_1, C_2\}\} : 2 : 1$	
n_3	$\{b, c\}$			2	$\emptyset : 2 : 1$	
	$\{b\}$	$\{c\}$		2	$\{\{C_1, C_2\}\} : 2 : 1$	$\emptyset : 3 : 1$
n_4	$\{b, c\}$			1	$\emptyset : 1 : 1$	
	$\{b\}$	$\{c\}$		2	$\emptyset : 2 : 1$	
n_5	$\{b, c\}$			3	$\emptyset : 3 : 1$	
	$\{b\}$	$\{c\}$		3	$\{\{C_1, C_2\}\} : 3 : 1$	$\emptyset : 4 : 1$
n_6	$\{b, c, f\}$			3		
	$\{b\}$	$\{c, f\}$		3		
	$\{b, c\}$	$\{f\}$		3	$\emptyset : 3 : 1$	
	$\{b, f\}$	$\{c\}$		3		
	$\{b\}$	$\{c\}$	$\{f\}$	3	$\{\{C_1, C_2\}\} : 3 : 1$	$\emptyset : 4 : 1$
n_7	$\{c, f\}$			3		
	$\{c\}$	$\{f\}$		3	$\emptyset : 3 : 2$	
n_8	$\{c, e, f\}$			3	$\emptyset : 3 : 2$	
	$\{c\}$	$\{e, f\}$		4	$\emptyset : 4 : 2$	
	$\{c, e\}$	$\{f\}$		4	$\emptyset : 4 : 2$	
	$\{c, f\}$	$\{e\}$		5		
	$\{c\}$	$\{e\}$	$\{f\}$	5	$\emptyset : 5 : 2$	

Fig. 2. Colorings of the tree decomposition of Example 2.

This lemma (together with Definition 4) naturally yields an algorithm to determine the size of an optimal solution by computing $\pi_n(\mathcal{C})$ for all colorings \mathcal{C} of all nodes n in the tree decomposition. We start by determining $\pi_n(\mathcal{C})$ for all colorings \mathcal{C} of leaf nodes n . For all remaining nodes $n \in T$, the values $\pi_n(\mathcal{C})$ of all colorings \mathcal{C} are computed in a bottom-up fashion along the tree decomposition. Finally, we obtain the cardinality of the minimum cuts as the minimal value $\pi_{n_{root}}(\mathcal{C})$ among the colorings \mathcal{C} at the root node n_{root} .

Example 4. We illustrate some aspects of this algorithm on our example problem using Figure 2. Consider, for instance, coloring $\mathcal{C} = \{\{b\}, \{c\}\}$ for node n_3 . We have two colorings for n_1 (the child node of n_3) which extend \mathcal{C} , namely, $\mathcal{C}' = \{\{b, d\}, \{c\}\}$ and $\mathcal{C}'' = \{\{b\}, \{c\}, \{d\}\}$. We know $\sigma_{n_1}(\mathcal{C}') = \pi_{n_1}(\mathcal{C}') = |\Gamma(\mathcal{C}')| = 2$, $\sigma_{n_1}(\mathcal{C}'') = \pi_{n_1}(\mathcal{C}'') = |\Gamma(\mathcal{C}'')| = 3$, and thus we select the smaller number and assign $\pi_{n_3}(\mathcal{C}) = 2$. For the branch node n_5 , consider coloring $\mathcal{C} = \{\{b\}, \{c\}\}$. In a bottom-up traversal, we already know $\pi_{n_3}(\mathcal{C}) = \pi_{n_4}(\mathcal{C}) = 2$, and we clearly have $|\Gamma(\mathcal{C})| = 1$. Thus, $\pi_{n_5}(\mathcal{C}) = 2 + 2 - 1 = 3$, which equals $\sigma_{n_5}(\mathcal{C})$ as expected.

Theorem 1. *Given an instance (G, H) of EMC and a tree decomposition of (G, H) of width w^* , the cardinality $|F|$ of the solutions $F \in \text{MinCuts}(G, H)$ can be computed in time $O(2^{2w^* \log w^*} \cdot \|(G, H)\|)$, assuming unit cost for arithmetic operations.*

The algorithm presented above allows us to compute a minimum cut F in a simple postprocessing step. This can be done by a recursive procedure which proceeds by a top-down traversal of the tree decomposition. This procedure takes as input a node n and a coloring \mathcal{C} at n and returns a set of edges (which are part of the cut F). Initially, we choose an arbitrary coloring \mathcal{C} with minimal value $\sigma_{n_{root}}(\mathcal{C})$ among the colorings for n_{root} . The recursion starts with the root node n_{root} and this coloring \mathcal{C} . This initial call to the recursion returns a minimum cut. Depending on the type of each node n , the following action is performed.

- (a) in case n is a (vI)-node, there is a unique coloring \mathcal{C}' for the child node n' of n , such that $\mathcal{C}' \leq \mathcal{C}$. We continue the recursion with n' and \mathcal{C}' . Let the return value for (n', \mathcal{C}') be Γ' . Then, the return value for (n, \mathcal{C}) is $\Gamma' \cup \Gamma(\mathcal{C})$;
- (b) in case n is a (vR)-node with child node n' , we choose a coloring \mathcal{C}' for n' with $\mathcal{C} \leq \mathcal{C}'$, such that $\pi_{n'}(\mathcal{C}') = \pi_n(\mathcal{C})$. We continue the recursion with n' and \mathcal{C}' and the return value for (n', \mathcal{C}') is also the return value for (n, \mathcal{C}) ;
- (c) in case n is a (B)-node with children n' and n'' , we continue the recursion for both children with n' and \mathcal{C} , resp. with n'' and \mathcal{C} . The return value for (n, \mathcal{C}) is the union of the return values of these two calls.
- (d) for a leaf node n , we just return $\Gamma(\mathcal{C})$.

Example 5. For our example problem, we thus have to start with the coloring $\mathcal{C} = \{\{c, e, f\}\}$ of n_8 . \mathcal{C} extends coloring $\{\{c, f\}\}$ of n_7 , which is an (R)-node. We have to choose a coloring of n_6 which extends $\{\{c, f\}\}$. Two such colorings exist, $\{\{b, c, f\}\}$ and $\{\{b\}, \{c, f\}\}$, and both have the same π -value as $\{\{c, f\}\}$. Let us select $\mathcal{C}' = \{\{b\}, \{c, f\}\}$. We then have to proceed with $\mathcal{C}'' = \{\{b\}, \{c\}\}$ at n_5 , which is a (B)-node. At the (B) node, we proceed as follows:

We first continue with \mathcal{C}'' in n_3 which is a (dR)-node. There are two colorings of n_1 which extend \mathcal{C}'' , namely $\{\{b, d\}, \{c\}\}$ and $\{\{b\}, \{c\}, \{d\}\}$. However, only the former has the same π -value as \mathcal{C}'' and is thus selected. Since we are now at a leaf node, we return the induced cut $\Gamma_1 = \{\{b, c\}, \{c, d\}\}$ of $\{\{b, d\}, \{c\}\}$.

Now we go into the second recursion with \mathcal{C}'' in n_4 which is an (aR)-node. There are two colorings for the child n_2 which extend \mathcal{C}'' , namely $\{\{a, c\}, \{b\}\}$ and $\{\{a\}, \{b\}, \{c\}\}$. Both have the same π -value, so we can use either of them. Let us select $\{\{a, c\}, \{b\}\}$; we return its induced cut $\Gamma_2 = \{\{a, b\}, \{b, c\}\}$.

Back at the branch node, we have to combine the return values and obtain $\Gamma = \Gamma_1 \cup \Gamma_2 = \{\{a, b\}, \{b, c\}, \{c, d\}\}$ which is the return value at n_6 . For the return value here, we observe $\Gamma \cup \Gamma(\mathcal{C}') = \Gamma$ and thus return Γ which is passed on unchanged by (R)-node n_7 . We compute the final return value at n_8 as $\Gamma \cup \Gamma(\mathcal{C}) = \Gamma \cup \emptyset = \Gamma$. Indeed, Γ is a minimum cut of our problem.

As this example shows, our postprocessing step would have yielded the same solution, if we had selected $\{\{a\}, \{b\}, \{c\}\}$ instead of $\{\{a, c\}, \{b\}\}$ at node n_2 . This is due to the fact that our data model does not guarantee that two distinct colorings $\mathcal{C}, \mathcal{C}'$ of some node n , satisfy $\mathcal{E}_n(\mathcal{C}) \cap \mathcal{E}_n(\mathcal{C}') = \emptyset$. This, in particular, causes problems for giving a general enumeration algorithm (it would yield duplicate results) and is also a serious obstacle for counting the number of solutions.

4 Counting and enumeration

In the previous section, the key objects to maintain during the bottom-up traversal of the tree decompositions were the colorings $\mathcal{C} \in \mathbf{C}(n)$ for each node $n \in T$. As we have seen, it is possible that two distinct colorings $\mathcal{C}_1, \mathcal{C}_2 \in \mathbf{C}(n)$ share some solutions, i.e. $\mathcal{E}_n(\mathcal{C}_1) \cap \mathcal{E}_n(\mathcal{C}_2) = \emptyset$ does not necessarily hold. The reason is, that we guarantee for two *distinct colors* $C_i, C_j \in \mathcal{C}$, only that the cut *disconnects* all vertices $v \in C_i$ from the vertices $w \in C_j$. No statement was made about the *connectedness of vertices contained in the same partition* C_i . We now modify our approach so as to guarantee this connectedness.

Definition 5. A parsimonious coloring (pcoloring, for short) over a set $W \subseteq V$ of vertices is a tuple (\mathcal{C}, Δ) , where $\mathcal{C} \in \mathbf{C}(W)$ and Δ is a set of unordered pairs $\{C_i, C_j\}$ of distinct colors $C_i, C_j \in \mathcal{C}$, such that for each $\{u, v\} \in H$ it holds that $\{C_i, C_j\} \in \Delta$, whenever $u \in C_i$ and $v \in C_j$. Δ is called the disconnection relation of the pcoloring $\mathcal{P} = (\mathcal{C}, \Delta)$. We shall use \mathcal{P}_c to identify the coloring \mathcal{C} and \mathcal{P}_d to identify the disconnection relation Δ . Moreover, we denote by $\mathbf{P}(W)$ the set of all pcolorings over W and write $\mathbf{P}(n)$ as a shorthand for $\mathbf{P}(\chi(n))$.

Given $\mathcal{P} \in \mathbf{P}(n)$, \mathcal{P}_c is simply a coloring of n . Ultimately, we are only interested in colorings \mathcal{P}_c , s.t. (in some extension of \mathcal{P}_c) the vertices in each color are indeed connected in the subgraph of G induced by the vertices $\chi(T_n)$. In contrast to the previous section, we now allow that two different colors C_i, C_j get *melted* to a single color during the bottom-up traversal of T ; this happens, e.g., at a (vI)-node if v has edges to some vertices $v_i \in C_i$ and $v_j \in C_j$ and if none of these edges is added to the cut. However, if for some terminal pair $\{h_1, h_2\} \in H$, we have $h_1 \in C_i$ and $h_2 \in C_j$, then these colors C_i, C_j must never get melted as the bottom-up traversal continues. The very purpose of the second component \mathcal{P}_d of \mathcal{P} is to keep track of such pairs which must not get melted. Clearly, for each coloring \mathcal{C} , there exists a relation Δ , s.t. (\mathcal{C}, Δ) is a pcoloring. Below, we overload the concepts from Definitions 2 and 3 for pcolorings.

Definition 6. Let $W \subseteq W' \subseteq V$, $\mathcal{P} \in \mathbf{P}(W)$ and $\mathcal{P}' \in \mathbf{P}(W')$. Then, $\mathcal{P} \leq \mathcal{P}'$, if $\mathcal{P}_c \leq \mathcal{P}'_c$ and $\mathcal{P}_d = \{\{C_i \cap W, C_j \cap W\} : \{C_i, C_j\} \in \mathcal{P}'_d\} \setminus \{\{\emptyset, \cdot\}, \{\cdot, \emptyset\}\}$.

For a node $n \in T$, let us denote by G_n the subgraph induced by the vertices $\chi(T_n)$, i.e. $G_n = (\chi(T_n), E_n)$ where $E_n = E \cap \{\{v, u\} : v, u \in \chi(T_n)\}$. Moreover, for a graph G_n , let $\kappa(G_n)$ denote the set of connected components of G_n , and, for a set of edges F , let $G_n \setminus F = (\chi(T_n), E_n \setminus F)$ as expected.

Definition 7. Let $\Delta(\mathcal{C}) = \{\{C_i, C_j\} \subseteq \mathcal{C} : \{u, v\} \in H, u \in C_i, v \in C_j\}$ for coloring \mathcal{C} . Given a node $n \in T$ and $\mathcal{P} \in \mathbf{P}(n)$, we define

$$\text{ext}_n(\mathcal{P}) = \{\mathcal{P}' \in \mathbf{P}(\chi(T_n)) : \mathcal{P} \leq \mathcal{P}', \kappa(G_n \setminus \Gamma(\mathcal{P}'_c)) = \mathcal{P}'_c, \mathcal{P}'_d = \Delta(\mathcal{P}'_c)\}.$$

\mathcal{P} is called valid (in n) if $\text{ext}_n(\mathcal{P}) \neq \emptyset$.

The remaining two concepts are now only defined for valid pcolorings.

$$\begin{aligned} \sigma_n(\mathcal{P}) &= \min\{|\Gamma(\mathcal{P}'_c)| : \mathcal{P}' \in \text{ext}_n(\mathcal{P})\} \\ \mathcal{E}_n(\mathcal{P}) &= \{\Gamma(\mathcal{P}'_c) : \mathcal{P}' \in \text{ext}_n(\mathcal{P}) \text{ s.t. } |\Gamma(\mathcal{P}'_c)| = \sigma_n(\mathcal{P})\}. \end{aligned}$$

The modified definition of extensions guarantees that the vertices in each color are indeed connected in the subgraph of G induced by the vertices $\chi(T_n)$ and that this connectedness is still fulfilled after removing the edges contained in the cut. This property is ensured by the condition $\kappa(G_n \setminus \Gamma(\mathcal{P}'_c)) = \mathcal{P}'_c$ in the definition of $\text{ext}_n(\mathcal{P})$. The condition $\mathcal{P}'_d = \Delta(\mathcal{P}'_c)$ ensures that \mathcal{P}'_d contains only disconnection pairs that are justified by some terminal pair in H . As a consequence, \mathcal{P}'_d is uniquely determined by \mathcal{P}'_c , which will be crucial for Lemma 4 below. Of course, $\text{ext}_n(\mathcal{P})$ may be empty. For instance, consider the coloring $\mathcal{C} = \{\{a, c\}, \{b\}\}$ at node n_2 of our running example. Since b is in a different color than a and c , both edges (a, b) and (b, c) are contained in the induced cut $\Gamma(\mathcal{C})$. But then $\{a, c\}$ is disconnected in the corresponding subgraph $G_{n_2} \setminus \Gamma(\mathcal{C})$. Hence, there exists no Δ to turn (\mathcal{C}, Δ) into a valid pcoloring. Actually, all rows in Figure 2 where the last two columns are empty contain colorings that cannot be part of a valid pcoloring. Analogously to Lemma 1, particular pcolorings for n_{root} characterize the minimum cuts of a given problem instance.

Lemma 3. *Let \mathbf{P}_{\min} be the set of all valid pcolorings \mathcal{P} of n_{root} , for which there exists no other valid pcoloring \mathcal{P}' of n_{root} , such that $\sigma_{n_{\text{root}}}(\mathcal{P}') < \sigma_{n_{\text{root}}}(\mathcal{P})$. Then, $\text{MinCuts}(G, H) = \bigcup_{\mathcal{P} \in \mathbf{P}_{\min}} \mathcal{E}_{n_{\text{root}}}(\mathcal{P})$.*

Lemma 4. *Given a node $n \in T$ and distinct valid pcolorings \mathcal{P}_1 and \mathcal{P}_2 of n , it holds that $\mathcal{E}_n(\mathcal{P}_1) \cap \mathcal{E}_n(\mathcal{P}_2) = \emptyset$.*

Next, we define two operations on pcolorings which we need for the bottom-up traversal of T . The *melting* operation is denoted as \mathcal{P}^S . It takes a pcoloring \mathcal{P} and some set S of vertices and melts all colors containing at least one vertex from S to a single color. This operation will be needed at (vI)-nodes where all colors containing some vertex v_i adjacent to v get melted into a single color (unless we add the edge connecting v and v_i to the cut). The second operation is the *composition* of two pcolorings $\mathcal{P}, \mathcal{P}'$, denoted as $\mathcal{P} \cup \mathcal{P}'$. Intuitively, the first component of $\mathcal{P} \cup \mathcal{P}'$ is obtained as the connected components of a graph of which we know that all vertices jointly occurring in some color of \mathcal{P} or \mathcal{P}' are connected. The composition operation will be needed at (B)-nodes. For both operations, the disconnection relation has to be adapted correspondingly. We first introduce the following notation for a pcoloring $\mathcal{P} \in \mathbf{P}(W)$ and a set $S \subseteq W \subseteq V$:

$$\mathcal{P}|_S = \{C \in \mathcal{P}_c : C \cap S \neq \emptyset\} \quad \text{and} \quad [\mathcal{P}, S] = \bigcup_{C \in \mathcal{P}|_S} C.$$

Moreover, for a vertex u we write $[\mathcal{P}, u]$ instead of $[\mathcal{P}, \{u\}]$ which simply denotes the color $C \in \mathcal{P}_c$ with $u \in C$.

Definition 8. *Let $W \subseteq V$, $\mathcal{P} \in \mathbf{P}(W)$ and $S \subseteq W$. Then $\mathcal{P}^S = (\mathcal{C}, \Delta)$ where $\mathcal{C} = \{[\mathcal{P}, S]\} \cup (\mathcal{P}_c \setminus \mathcal{P}|_S)$ and*

$$\Delta = \left\{ \{[\mathcal{P}, S], C'\} : C' \in \mathcal{P}_c \setminus \mathcal{P}|_S, C \in \mathcal{P}|_S, \{C, C'\} \in \mathcal{P}_d \right\} \cup \left\{ \{C, C'\} : C, C' \in \mathcal{C}, \{C, C'\} \in \mathcal{P}_d \right\}.$$

Definition 9. *Let $W \subseteq V$, $\mathcal{P}, \mathcal{P}' \in \mathbf{P}(W)$, and for each $w \in W$, let C_w be the smallest set such that $w \in C_w$ and, for each $u \in C_w$, $[\mathcal{P}, u] \cup [\mathcal{P}', u] \subseteq C_w$. We define $\mathcal{P} \cup \mathcal{P}' = (\mathcal{C}, \Delta)$ where $\mathcal{C} = \{C_w : w \in W\}$ and*

$$\Delta = \left\{ \{C_1, C_2\} : C_1, C_2 \in \mathcal{C}, C_1 \neq C_2, C'_1 \subseteq C_1, C'_2 \subseteq C_2, \{C'_1, C'_2\} \in \mathcal{P}_d \cup \mathcal{P}'_d \right\}.$$

For $S = \emptyset$, or a singleton S , we get $\mathcal{P}^S = \mathcal{P}$ as expected. Note that \mathcal{P}^S is possibly *not* a pcoloring. This is the case if $h \subseteq [\mathcal{P}, S]$ for some terminal pair $h \in H$. Moreover, for a *valid* pcoloring \mathcal{P} , the pcoloring \mathcal{P}^S may be *invalid*, if $(C_1 \cup C_2) \subseteq [\mathcal{P}, S]$ for some pair $\{C_1, C_2\} \in \mathcal{P}_d$. Likewise, $\mathcal{P} \cup \mathcal{P}'$ may be not a pcoloring. This is the case if a pair $\{h_1, h_2\}$ of terminal vertices ends up in a joint color. Just consider the simple situation that \mathcal{P}_c contains a color $\{h_1, v\}$ and \mathcal{P}'_c contains a color $\{h_2, v\}$ for some vertex v . It may also happen that two valid pcolorings \mathcal{P} and \mathcal{P}' lead to an invalid pcoloring $\mathcal{P} \cup \mathcal{P}'$ if $\mathcal{P} \cup \mathcal{P}'$ contains a color C , s.t. $C_1 \cup C_2 \subseteq C$ for some pair $\{C_1, C_2\}$ in \mathcal{P}_d or \mathcal{P}'_d .

Next, we define relations between pcolorings over adjacent nodes in the tree decomposition. These relations will be used to traverse the tree decomposition.

Definition 10. *Let $n \in T$, $\mathcal{P} \in \mathbf{P}(n)$, $\mathcal{P}' \in \mathbf{P}(n')$, and $\mathcal{P}'' \in \mathbf{P}(n'')$. Then, the following pairs are contained in relation \prec_n .*

1. $\mathcal{P}' \prec_n \mathcal{P}$, if n is a (vI)-node with child n' , and there exists a set $S \subseteq \chi(n') \cap \{u: \{v, u\} \in E\}$, such that $(\mathcal{P}')^S \leq \mathcal{P}$, $[\mathcal{P}', S] \cup \{v\} = [\mathcal{P}, v]$, and for each $\{C', D'\} \in (\mathcal{P}')_d$, there is a $\{C, D\} \in \mathcal{P}_d$ with $C' \subseteq C$ and $D' \subseteq D$.
2. $\mathcal{P}' \prec_n \mathcal{P}$, if n is an (R)-node with child n' , and $\mathcal{P} \leq \mathcal{P}'$.
3. $(\mathcal{P}', \mathcal{P}'') \prec_n \mathcal{P}$, if n has two children n', n'' , $\mathcal{P} = \mathcal{P}' \cup \mathcal{P}''$, and for each $\{C', D'\} \in (\mathcal{P}')_d \cup (\mathcal{P}'')_d$, there is a $\{C, D\} \in \mathcal{P}_d$ with $C' \subseteq C$ and $D' \subseteq D$.

As already mentioned above, some colorings at a child node may get melted at (I)-nodes or (B)-nodes. For a (vI)-node, S in the above definition denotes an arbitrarily chosen subset of all vertices adjacent to v , s.t. all colors containing a vertex in S are then melted into a single color C_i on the transition from \mathcal{P}' to \mathcal{P} . The neighbors of v not selected in S may end up in a different color C_j . Implicitly, the edges connecting v with these neighbors are thus added to the cut. For (B)-nodes, the condition $\mathcal{P} = \mathcal{P}' \cup \mathcal{P}''$ forces $\mathcal{P}' \cup \mathcal{P}''$ to be a pcoloring. Moreover, the existence of $\{C', D'\} \in \mathcal{P}_d$ with $C \subseteq C'$ and $D \subseteq D'$ makes sure that no “disconnected pair” in either \mathcal{P}'_d or \mathcal{P}''_d gets melted into a single color.

Lemma 5. *If $\mathcal{P}' \in \mathbf{P}(n')$ is valid in n' (and $\mathcal{P}'' \in \mathbf{P}(n'')$ is valid in n''), then each $\mathcal{P} \in \mathbf{P}(n)$ with $\mathcal{P}' \prec_n \mathcal{P}$ (resp. $(\mathcal{P}', \mathcal{P}'') \prec_n \mathcal{P}$) is valid in n . If $\mathcal{P} \in \mathbf{P}(n)$ is valid in n , then there exists $\mathcal{P}' \in \mathbf{P}(n')$ (and $\mathcal{P}'' \in \mathbf{P}(n'')$) with $\mathcal{P}' \prec_n \mathcal{P}$ (resp. $(\mathcal{P}', \mathcal{P}'') \prec_n \mathcal{P}$), s.t. \mathcal{P}' is valid in n' (and \mathcal{P}'' is valid in n'').*

Similarly to Section 3, we can compute $\sigma_n(\mathcal{P})$ for valid pcolorings \mathcal{P} by a bottom-up traversal of T . The cases of (vI) and (B) nodes are slightly more complicated, since different pcolorings might collapse into the same pcoloring.

Definition 11. *For a valid pcoloring $\mathcal{P} \in \mathbf{P}(n)$ and $n \in T$ (possibly with children n', n''), we define $\pi_n(\mathcal{P})$ recursively, depending on the node type of n .*

- (L)-node: $\pi_n(\mathcal{P}) = |\Gamma(\mathcal{P}_c)|$;
- (vR)-node: $\pi_n(\mathcal{P}) = \min\{\pi_{n'}(\mathcal{P}'): \mathcal{P}' \prec_n \mathcal{P}\}$;
- (vI)-node: $\pi_n(\mathcal{P}) = \min\{\pi_{n'}(\mathcal{P}'): \mathcal{P}' \prec_n \mathcal{P}\} + |\{u: \{u, v\} \in \Gamma(\mathcal{P}_c)\}|$;
- (B)-node: $\pi_n(\mathcal{P}) = \min\{\pi_{n'}(\mathcal{P}') + \pi_{n''}(\mathcal{P}''): (\mathcal{P}', \mathcal{P}'') \prec_n \mathcal{P}\} - |\Gamma(\mathcal{P}_c)|$.

Lemma 6. *For any node $n \in T$ and any valid pcoloring \mathcal{P} of n , $\sigma_n(\mathcal{P}) = \pi_n(\mathcal{P})$.*

We are now ready to construct an algorithm to count the solutions of an EMC instance. One further notation is required: We restrict the relation \prec_n on pcolorings to a relation \prec_n^{\min} , s.t. $\mathcal{P}' \prec_n^{\min} \mathcal{P}$ (resp. $(\mathcal{P}', \mathcal{P}'') \prec_n^{\min} \mathcal{P}$) only holds if $\pi_n(\mathcal{P})$ is determined by $\pi_{n'}(\mathcal{P}')$ (resp. by $\pi_{n'}(\mathcal{P}')$ and $\pi_{n''}(\mathcal{P}'')$) in Definition 11. I.e., $\mathcal{P}' \prec_n^{\min} \mathcal{P}$ holds iff $\mathcal{P}' \prec_n \mathcal{P}$ and either $\pi_n(\mathcal{P}) = \min\{\pi_{n'}(\mathcal{P}'): \mathcal{P}' \prec_n \mathcal{P}\}$ (for (R)-nodes) or $\pi_n(\mathcal{P}) = \min\{\pi_{n'}(\mathcal{P}'): \mathcal{P}' \prec_n \mathcal{P}\} + |\{u: \{u, v\} \in \Gamma(\mathcal{P}_c)\}|$ (for (I)-nodes) holds. Likewise, $(\mathcal{P}', \mathcal{P}'') \prec_n^{\min} \mathcal{P}$ holds iff $(\mathcal{P}', \mathcal{P}'') \prec_n \mathcal{P}$ and $\pi_n(\mathcal{P}) = \min\{\pi_{n'}(\mathcal{P}') + \pi_{n''}(\mathcal{P}''): (\mathcal{P}', \mathcal{P}'') \prec_n \mathcal{P}\} - |\Gamma(\mathcal{P}_c)|$.

Definition 12. For $\mathcal{P} \in \mathbf{P}(n)$ and $n \in T$ (possibly with children n', n''), let

$$\#_n(\mathcal{P}) = \begin{cases} 1 & n \text{ is (L)-node} \\ \sum_{\mathcal{P}' \prec_n^{\min} \mathcal{P}} \#_{n'}(\mathcal{P}') & n \text{ is (I)- or (R)-node} \\ \sum_{(\mathcal{P}', \mathcal{P}'') \prec_n^{\min} \mathcal{P}} \#_{n'}(\mathcal{P}') \cdot \#_{n''}(\mathcal{P}'') & n \text{ is (B)-node} \end{cases}$$

Lemma 7. For each $n \in T$ and each valid pcoloring \mathcal{P} of n , $|\mathcal{E}_n(\mathcal{P})| = \#_n(\mathcal{P})$.

By combining Lemmas 3, 4, and 7, we immediately get the following theorem.

Theorem 2. Let \mathbf{P}_{\min} be the set of all valid pcolorings \mathcal{P} of n_{root} , for which there exists no other pcoloring \mathcal{P}' of n_{root} , such that $\sigma_{n_{\text{root}}}(\mathcal{P}') < \sigma_{n_{\text{root}}}(\mathcal{P})$. Then, $|\text{MinCuts}(G, H)| = \sum_{\mathcal{P} \in \mathbf{P}_{\min}} \#_{n_{\text{root}}}(\mathcal{P})$.

Example 6. We revisit the instance of the EMC problem from Example 2. In Figure 2, the result of computing $\sigma_n(\mathcal{P})$ and $\#_n(\mathcal{P})$ for all valid pcolorings \mathcal{P} at all nodes n is shown. Only the rows with an entry in one of the last two columns correspond to a valid pcoloring \mathcal{P} . In this case, the sets in the second column (with heading C_1, C_2, C_3) and the coloring \mathcal{P}_c and the column labeled $\Delta_1 : \sigma : \#$ contains the set of “disconnected pairs” \mathcal{P}_d together with the size of the corresponding cuts in $\mathcal{E}_n(\mathcal{P})$ and the number $\#_n(\mathcal{P})$ of such cuts. Some colorings \mathcal{P}_c admit two possible sets of “disconnected pairs” in order to form a valid pcoloring. In these cases, the last column contains the second possibility – again together with the corresponding size of the cuts and the counter $\#_n(\mathcal{P})$. The minimum cuts correspond to the valid pcolorings at the root node n_s with minimal value of $\sigma_{n_s}(\mathcal{P})$. In our example, the only such pcoloring is $\mathcal{P} = (\{\{c, e, f\}\}, \emptyset)$. Thereby the cuts corresponding to \mathcal{P} consist of three edges and there are, in total, two minimum cuts (see Example 2).

We conclude this section by outlining an algorithm for enumerating all solutions with fixed-parameter linear delay by extending the computation of a single solution in the previous section. The basic idea is to traverse the tree decomposition in top-down direction several times – namely once for each solution. In the recursive algorithm described in Section 3, we made arbitrary choices at the root node (for selecting one coloring with optimal value $\sigma_{n_{\text{root}}}(\mathcal{C})$) and at internal nodes (e.g., at an (R)-node n , for selecting some coloring \mathcal{C}' at the child node n' with $\mathcal{C} \leq \mathcal{C}'$). For enumerating all solutions, we have to maintain lists to keep track of all possible choices at each node in the tree decomposition. These lists allow us to iterate through all possible choices. Hence, at the root, we iterate

through all *valid pcolorings* \mathcal{P} of n_{root} with minimal value $\sigma_{n_{root}}(\mathcal{P})$. Likewise, for a valid pcoloring \mathcal{P} at some (R)- or (I)-node n , we eventually have to process all pcolorings \mathcal{P}' at the child node n' of n , s.t. $\mathcal{P}' \prec_n^{\min} \mathcal{P}$. Finally, also for a valid pcoloring \mathcal{P} at some (B)-node n , we eventually have to process all pairs $(\mathcal{P}', \mathcal{P}'')$ of pcolorings at the child nodes n' and n'' of n , s.t. $(\mathcal{P}', \mathcal{P}'') \prec_n^{\min} \mathcal{P}$. Only at (L)-nodes, no iterating of several possible pcolorings is required. In [16], we state the full algorithm, which crucially depends on Lemma 4, guaranteeing that no duplicates are produced. Note that this algorithm could also be used for the optimization problem, but this would result in a loss of efficiency.

Theorem 3. *Given an instance (G, H) of the EMC problem together with a tree decomposition \mathcal{T} of (G, H) having width w^* . Assuming unit cost for arithmetic operations, $O(2^{2((w^*)^2 + w^* \log w^*)} \cdot \|(G, H)\|)$ bounds the time to compute $|\text{MinCuts}(G, H)|$ as well as the delay to enumerate $\text{MinCuts}(G, H)$.*

5 Conclusion

We have presented novel algorithms for the optimization and counting problem of EDGE MULTICUT. Moreover, we have outlined how the counting algorithm can be extended to an enumeration algorithm. It is now straightforward to adapt these algorithms to the (RESTRICTED or UNRESTRICTED) VERTEX MULTICUT problem. Indeed, for the optimization problem, we can proceed as in Section 3. The main modification required is to allow colorings where some vertices possibly do not get a color assigned. The intended meaning is that colorless vertices are exactly those which are part of the cut. The pcolorings introduced in Section 4 for the counting and enumeration problem have to be adapted analogously. The details of these algorithms are left for future work. Further extending these algorithms to the weighted multicut problems is even simpler: Rather than maintaining the *cardinality* π_n as in Definitions 4 and 11, we would now have to keep track of the *total weight* of a (p)coloring.

Note that the upper bounds on the constants of our algorithms (in Theorems 1 and 3) are obtained by very coarse estimates – assuming straightforward methods for storing and manipulating partitions, etc. For future work, we plan to implement our multicut algorithms – using more sophisticated data structures and algorithms – which should also improve the upper bounds.

References

1. Costa, M.C., Létocart, L., Roupin, F.: Minimal multicut and maximal integer multiflow: A survey. *European Journal of Operational Research* **162** (2005) 55–69
2. Călinescu, G., Fernandes, C.G., Reed, B.A.: Multicuts in unweighted graphs and digraphs with bounded degree and bounded tree-width. *J. Alg.* **48** (2003) 333–359
3. Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M.: The complexity of multiterminal cuts. *SIAM J. Comput.* **23** (1994) 864–894
4. Garg, N., Vazirani, V.V., Yannakakis, M.: Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica* **18** (1997) 3–20

5. Bentz, C.: A simple algorithm for multicuts in planar graphs with outer terminals. *Discrete Applied Mathematics* **157** (2009) 1959–1964
6. Flum, J., Grohe, M.: *Parameterized Complexity Theory*. Springer (2006)
7. Bousquet, N., Daligault, J., Thomassé, S., Yeo, A.: A polynomial kernel for multicut in trees. In: Proc. STACS'09. Volume 3 of LIPIcs. (2009) 183–194
8. Marx, D., Razgon, I.: Constant ratio fixed-parameter approximation of the edge multicut problem. In: Proc. ESA'09. Vol. 5757 of LNCS., Springer (2009) 647–658
9. Marx, D.: Parameterized graph separation problems. *Theor. Comput. Sci.* **351** (2006) 394–406
10. Xiao, M.: Simple and improved parameterized algorithms for multiterminal cuts (to appear). *Theory Comput. Syst.* (2010)
11. Guo, J., Hüffner, F., Kenar, E., Niedermeier, R., Uhlmann, J.: Complexity and exact algorithms for vertex multicut in interval and bounded treewidth graphs. *European Journal of Operational Research* **186** (2008) 542–553
12. Bentz, C.: On the complexity of the multicut problem in bounded tree-width graphs and digraphs. *Discrete Applied Mathematics* **156** (2008) 1908–1917
13. Gottlob, G., Lee, S.T.: A logical approach to multicut problems. *Inf. Process. Lett.* **103** (2007) 136–141
14. Courcelle, B.: Graph rewriting: An algebraic and logic approach. In: *Handbook of Theor. Comp. Sci.*, Volume B. Elsevier Science Publishers (1990) 193–242
15. Arnborg, S., Lagergren, J., Seese, D.: Easy problems for tree-decomposable graphs. *J. Algorithms* **12** (1991) 308–340
16. Pichler, R., Rümmele, S., Woltran, S.: Multicut algorithms via tree decompositions. Technical Report DBAI-TR-2010-67, Technische Universität Wien (2010)
17. Kloks, T.: *Treewidth: Computations and Approximations*. Springer, Berlin (1994)
18. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* **25** (1996) 1305–1317
19. van den Eijkhof, F., Bodlaender, H.L., Koster, A.M.C.A.: Safe reduction rules for weighted treewidth. *Algorithmica* **47** (2007) 139–158
20. Bodlaender, H.L., Koster, A.M.C.A.: Combinatorial optimization on graphs of bounded treewidth. *Comput. J.* **51** (2008) 255–269

Appendix

A Proofs

A.1 Proof of Lemma 1

\subseteq -direction. Let $F \in \text{MinCuts}(G, H)$. We construct coloring \mathcal{C}' of V by assigning a color $C \in \mathcal{C}'$ to each connected component of $(V, E \setminus F)$. Note that this construction ensures $\Gamma(\mathcal{C}') = F$. Now consider the coloring \mathcal{C} of n_{root} , for which $\mathcal{C} \leq \mathcal{C}'$. By Definition 3, $\mathcal{C}' \in \text{ext}_{n_{\text{root}}}(\mathcal{C})$. By Proposition 1, we know that for each $\mathcal{C}'' \in \mathbf{C}(V)$, $\Gamma(\mathcal{C}'') \in \text{Cuts}(G, H)$. Since $\Gamma(\mathcal{C}')$ is a minimum cut for (G, H) , we obtain (i) $\sigma_{n_{\text{root}}}(\mathcal{C}) = |\Gamma(\mathcal{C}')| = |F|$, and thus, $\Gamma(\mathcal{C}') \in \mathcal{E}_{n_{\text{root}}}(\mathcal{C})$; and (ii) $\mathcal{C} \in \mathbf{C}_{\text{min}}$. Hence $F \in \bigcup_{\mathcal{C} \in \mathbf{C}_{\text{min}}} \mathcal{E}_{n_{\text{root}}}(\mathcal{C})$.

\supseteq -direction. Let $\mathcal{C} \in \mathbf{C}_{\text{min}}$ and $F \in \mathcal{E}_{n_{\text{root}}}(\mathcal{C})$. Then by Definition 3, there exists $\mathcal{C}' \in \text{ext}_{n_{\text{root}}}(\mathcal{C})$, s.t. $F = \Gamma(\mathcal{C}')$ and $\sigma_{n_{\text{root}}}(\mathcal{C}) = |F|$. Since $\mathcal{C}' \in \mathbf{C}(\chi(T_{n_{\text{root}}})) = \mathbf{C}(V)$, it follows from Proposition 1 that $F \in \text{Cuts}(G, H)$. We have to show that $F \in \text{MinCuts}(G, H)$. Towards a contradiction, suppose the latter is not the case, i.e., there exists $F' \in \text{MinCuts}(G, H)$ with $|F'| < |F|$. By the same construction as in the \subseteq -direction, we get a coloring \mathcal{C}'' for n_{root} with $\sigma_{n_{\text{root}}}(\mathcal{C}'') = |F'|$. But then $\sigma_{n_{\text{root}}}(\mathcal{C}'') < \sigma_{n_{\text{root}}}(\mathcal{C})$ which contradicts $\mathcal{C} \in \mathbf{C}_{\text{min}}$.

A.2 Proof of Lemma 2

We show this by structural induction in bottom-up direction over normalized tree decompositions. If n is an (L) node, the statement is trivial. Let n be a (vR) node with child n' . By the induction hypothesis, for any $\mathcal{C}' \in \mathbf{C}(n')$, $\sigma_{n'}(\mathcal{C}') = \pi_{n'}(\mathcal{C}')$. Let \mathcal{C}' be one of the colorings in $\mathbf{C}(n')$ with $\mathcal{C} \leq \mathcal{C}'$ for which $\sigma_{n'}(\mathcal{C}')$ is the minimum. We have to show that $\sigma_n(\mathcal{C}) = \sigma_{n'}(\mathcal{C}')$. By Definition 3 there exists $\mathcal{C}'_e \in \text{ext}_{n'}(\mathcal{C}')$ with $\sigma_{n'}(\mathcal{C}') = |\Gamma(\mathcal{C}'_e)|$. Furthermore $\mathcal{C}'_e \in \mathbf{C}(\chi(T_{n'}))$ and $\mathcal{C}' \leq \mathcal{C}'_e$. Note that $\mathbf{C}(\chi(T_{n'})) = \mathbf{C}(\chi(T_n))$ and $\mathcal{C} \leq \mathcal{C}'_e$. Therefore $\mathcal{C}'_e \in \text{ext}_n(\mathcal{C})$ and hence $\sigma_n(\mathcal{C}) \leq |\Gamma(\mathcal{C}'_e)| = \sigma_{n'}(\mathcal{C}')$. Assume $\sigma_n(\mathcal{C}) < \sigma_{n'}(\mathcal{C}')$. This means there exists $\mathcal{C}_e \in \text{ext}_n(\mathcal{C})$ with $\sigma_n(\mathcal{C}) = |\Gamma(\mathcal{C}_e)|$. Moreover $\mathcal{C}_e \in \mathbf{C}(\chi(T_n))$ and $\mathcal{C} \leq \mathcal{C}_e$. Now take $\mathcal{C}'' \in \mathbf{C}(n')$ with $\mathcal{C}'' \leq \mathcal{C}_e$. Note that $\mathcal{C} \leq \mathcal{C}''$. Since $\mathcal{C}_e \in \text{ext}_{n'}(\mathcal{C}'')$, $\sigma_{n'}(\mathcal{C}'') \leq |\Gamma(\mathcal{C}_e)| = \sigma_n(\mathcal{C}) < \sigma_{n'}(\mathcal{C}')$. This contradicts the assumption for \mathcal{C}' and therefore $\sigma_n(\mathcal{C}) = \sigma_{n'}(\mathcal{C}')$. The cases where n is an (I) or a (B) node, can be shown analogously.

A.3 Proof of Theorem 1

We can compute $\pi_n(\mathcal{C})$ for every coloring \mathcal{C} at every node n in the tree decomposition \mathcal{T} by a single bottom-up traversal of \mathcal{T} and by applying the inductive definition of $\pi_n(\mathcal{C})$ according to Definition 4. Let $w = w^* + 1$. At every node n , there are at most $B(w)$ partitions of the vertices in $\chi(n)$, where $B(w)$ denotes the w th Bell number. In what follows, we will use $w!$ as a rough upper bound of $B(w)$. Hence, also the number of colorings \mathcal{C} at each node n is bounded by $w!$. We assume no sophisticated data structure for storing the colorings at each node n (e.g., a linked list is fine).

In order to compute $\pi_n(\mathcal{C})$ for an (L)-node, we just enumerate all possible partitions of $\chi(n)$ and check for each partition, if it is a coloring. The time required for each such check is clearly bounded by some polynomial $p(w)$, which is independent of the input size $\|(G, H)\|$, provided that the adjacency of two vertices in a bag can be tested in constant time (e.g., if the edges E in G and the pairs H are given by an adjacency matrix). The overall cost for this step is thus $O(w! \cdot p(w))$ in case of (L)-nodes.

For an (R)-node or (I)-node n with child node n' , we proceed in a nested loop over all possible colorings \mathcal{C} at n and all colorings \mathcal{C}' at n' . In total, we thus have to process at most $w! \cdot w!$ pairs of colorings. The action performed for each such pair again fits into time $O(p(w))$ for some polynomial $p(w)$, and is independent of the input size $\|(G, H)\|$. E.g., for a (vI)-node, we have to check if $\mathcal{C}' \leq \mathcal{C}$ holds. If this is the case, then we set $\pi_n(\mathcal{C}) = \pi_{n'}(\mathcal{C}') + |\{u: \{u, v\} \in \Gamma(\mathcal{C})\}|$.

Finally, for a (B)-node n with child nodes n' and n'' , we proceed in a nested loop over all possible colorings \mathcal{C}' at n' and all colorings \mathcal{C}'' at n'' . Again, we thus have to process at most $w! \cdot w!$ pairs of colorings, and the time required for processing each such pair fits into time $O(p(w))$ for some polynomial $p(w)$, and is independent of the input size $\|(G, H)\|$.

In summary, the effort required for processing each node n in \mathcal{T} is bounded by $O(w! \cdot w! \cdot p(w))$ for some polynomial $p(w)$. Note that the number of nodes is bounded by $O(\|(G, H)\|)$ and that $w! \cdot w! \cdot p(w) = O(2^k)$ with $k = 2 \cdot w \cdot \log(w^*)$. Thus, the desired time bound $O(2^k \cdot \|(G, H)\|)$ of the theorem is proved.

A.4 Proof of Lemma 3

The proof is almost identical to the one of Lemma 1. Instead of Definition 3 one has to use Definition 7 and for the \subseteq -direction, the construction of a coloring is done as follows. Given $F \in \text{MinCuts}(G, H)$, we construct a pcoloring \mathcal{P}' of V by assigning a color $C \in \mathcal{P}'_c$ to each connected component of $(V, E \setminus F)$. Furthermore, we set $\mathcal{P}'_d = \{\{C_i, C_j\} : C_i, C_j \in \mathcal{P}'_c, \{u, v\} \in H, u \in C_i, v \in C_j\}$.

A.5 Proof of Lemma 4

Assume $\mathcal{E}_n(\mathcal{P}_1) \cap \mathcal{E}_n(\mathcal{P}_2) \neq \emptyset$ and $\mathcal{P}_1 \neq \mathcal{P}_2$. Then there exist pcolorings $\mathcal{P}'_1 \in \text{ext}_n(\mathcal{P}_1)$ and $\mathcal{P}'_2 \in \text{ext}_n(\mathcal{P}_2)$, s.t. $\Gamma(\mathcal{P}'_{1c}) \in \mathcal{E}_n(\mathcal{P}_1)$, $\Gamma(\mathcal{P}'_{2c}) \in \mathcal{E}_n(\mathcal{P}_2)$ and $\Gamma(\mathcal{P}'_{1c}) = \Gamma(\mathcal{P}'_{2c})$. Therefore $\mathcal{P}'_{1c} = \kappa(G_n \setminus \Gamma(\mathcal{P}'_{1c})) = \kappa(G_n \setminus \Gamma(\mathcal{P}'_{2c})) = \mathcal{P}'_{2c}$. Moreover, $\mathcal{P}'_{1d} = \Delta(\mathcal{P}'_{1c}) = \Delta(\mathcal{P}'_{2c}) = \mathcal{P}'_{2d}$. But then, it follows from $\mathcal{P}_1 \leq \mathcal{P}'_1$ and $\mathcal{P}_2 \leq \mathcal{P}'_2$, that $\mathcal{P}_1 = \mathcal{P}_2$, which contradicts the assumption.

A.6 Proof of Lemma 5

We demonstrate this for (R) nodes. The other node types are shown analogously. Assume node $n \in T$ is of type (R) having child n' and assume $\mathcal{P}' \in \mathbf{P}(n')$ is valid in n' . Then there exists $\mathcal{P}'' \in \text{ext}_{n'}(\mathcal{P}')$ with $\mathcal{P}'' \in \mathbf{P}(\chi(T_{n'}))$, $\mathcal{P}' \leq \mathcal{P}''$, $\kappa(G_{n'} \setminus \Gamma(\mathcal{P}''_c)) = \mathcal{P}''_c$ and $\mathcal{P}''_d = \Delta(\mathcal{P}''_c)$. Since there exists exactly one $\mathcal{P} \in \mathbf{P}(n)$, s.t. $\mathcal{P} \leq \mathcal{P}'$, we have to show that \mathcal{P} is valid in n . From $\mathcal{P} \leq \mathcal{P}'$ and $\mathcal{P}' \leq \mathcal{P}''$

it follows that $\mathcal{P} \leq \mathcal{P}''$. Note that $\chi(T_n) = \chi(T_{n'})$ and $G_n = G_{n'}$. Therefore $\mathcal{P}'' \in \text{ext}_n(\mathcal{P})$ and \mathcal{P} is valid in n .

Now assume $\mathcal{P} \in \mathbf{P}(n)$ is valid in n . Then there exists $\mathcal{P}'' \in \text{ext}_n(\mathcal{P})$. We have to show, that there exists $\mathcal{P}' \in \mathbf{P}(n')$ that is valid in n' . Choose \mathcal{P}' , s.t. $\mathcal{P}' \leq \mathcal{P}''$. Since $\chi(T_n) = \chi(T_{n'})$ and $G_n = G_{n'}$, we have $\mathcal{P}'' \in \text{ext}_{n'}(\mathcal{P}')$ and therefore \mathcal{P}' is valid in n' .

A.7 Proof of Lemma 6

We show this by structural induction in bottom-up direction over normalized tree decompositions. If n is an (L) node, the statement is trivial. Let n be a (vR) node with child n' . By the induction hypothesis, for any $\mathcal{P}' \in \mathbf{P}(n')$, $\sigma_{n'}(\mathcal{P}') = \pi_{n'}(\mathcal{P}')$. Let \mathcal{P}' be one of the valid pcolorings in $\mathbf{P}(n')$ with $\mathcal{P} \leq \mathcal{P}'$ for which $\sigma_{n'}(\mathcal{P}')$ is the minimum. We have to show that $\sigma_n(\mathcal{P}) = \sigma_{n'}(\mathcal{P}')$. By Definition 7, there exists $\mathcal{P}'' \in \text{ext}_{n'}(\mathcal{P}')$ with $\sigma_{n'}(\mathcal{P}') = |\Gamma(\mathcal{P}''_c)|$. Furthermore $\mathcal{P}'' \in \mathbf{P}(\chi(T_{n'}))$, $\mathcal{P}' \leq \mathcal{P}''$, $\kappa(G_{n'} \setminus \Gamma(\mathcal{P}''_c)) = \mathcal{P}''_c$, and $\mathcal{P}''_d = \Delta(\mathcal{P}''_c)$. Note that $\mathbf{P}(\chi(T_{n'})) = \mathbf{P}(\chi(T_n))$, $G_{n'} = G_n$, and $\mathcal{P} \leq \mathcal{P}''$. Therefore $\mathcal{P}'' \in \text{ext}_n(\mathcal{P})$ and hence $\sigma_n(\mathcal{P}) \leq |\Gamma(\mathcal{P}''_c)| = \sigma_{n'}(\mathcal{P}')$. Assume $\sigma_n(\mathcal{P}) < \sigma_{n'}(\mathcal{P}')$. This means that there exists $\mathcal{P}''' \in \text{ext}_n(\mathcal{P})$ with $\sigma_n(\mathcal{P}) = |\Gamma(\mathcal{P}'''_c)|$. Now take the pcoloring $\mathcal{P}^{(4)}$ of n' with $\mathcal{P}^{(4)} \leq \mathcal{P}'''$. Note that $\mathcal{P} \leq \mathcal{P}^{(4)}$. Since $\mathcal{P}''' \in \text{ext}_{n'}(\mathcal{P}^{(4)})$, $\sigma_{n'}(\mathcal{P}^{(4)}) \leq |\Gamma(\mathcal{P}'''_c)| = \sigma_n(\mathcal{P}) < \sigma_{n'}(\mathcal{P}')$. This contradicts the assumption for \mathcal{P}' and therefore $\sigma_n(\mathcal{P}) = \sigma_{n'}(\mathcal{P}')$. The cases where n is an (I) or a (B) node, can be shown analogously.

A.8 Proof of Lemma 7

We proceed by structural induction over normalized tree decompositions. First, suppose that \mathcal{P} is a pcoloring at some (L)-node n . By Definition 7, $\text{ext}_n(\mathcal{P})$ contains precisely one element, namely \mathcal{P} itself. Likewise, $|\mathcal{E}_n(\mathcal{P})| = 1$. Hence, $|\mathcal{E}_n(\mathcal{P})| = \#_n(\mathcal{P}) = 1$ by Definition 12.

For an (R)-node n with child n' , we have $\mathcal{E}_n(\mathcal{P}) = \bigcup_{\mathcal{P}' \prec_n^{\min} \mathcal{P}} \mathcal{E}_{n'}(\mathcal{P}')$. This is a disjoint union by Lemma 4. Hence, by the induction hypothesis, $\mathcal{E}_n(\mathcal{P}) = \#_n(\mathcal{P}) = \sum_{\mathcal{P}' \prec_n^{\min} \mathcal{P}} \#_{n'}(\mathcal{P}')$ holds.

For a (vI)-node n with child n' , we have

$$\mathcal{E}_n(\mathcal{P}) = \bigcup_{\mathcal{P}' \prec_n^{\min} \mathcal{P}} \{\Gamma' \cup \Gamma(\mathcal{P}_c) : \Gamma' \in \mathcal{E}_{n'}(\mathcal{P}')\}.$$

By the induction hypothesis, it suffices to show that this is a disjoint union. By Lemma 4, any two sets $\Gamma'_1 \in \mathcal{E}_{n'}(\mathcal{P}'_1)$ and $\Gamma'_2 \in \mathcal{E}_{n'}(\mathcal{P}'_2)$ for two distinct pcolorings \mathcal{P}'_1 and \mathcal{P}'_2 at n' are distinct. Moreover, every set Γ' contains only edges in the subgraph of G induced by the vertices in $\chi(T_{n'})$. On the other hand, in every set $\Gamma' \cup \Gamma(\mathcal{P}_c) \in \mathcal{E}_n(\mathcal{P})$, all edges added to Γ' contain the vertex v as one endpoint. Hence, for any two sets $\Gamma'_1 \in \mathcal{E}_{n'}(\mathcal{P}'_1)$ and $\Gamma'_2 \in \mathcal{E}_{n'}(\mathcal{P}'_2)$, also $\Gamma'_1 \cup \Gamma(\mathcal{P}_c)$ and $\Gamma'_2 \cup \Gamma(\mathcal{P}_c)$ are distinct. But then, the union in the above definition of $\mathcal{E}_n(\mathcal{P})$ is indeed disjoint.

For a (B)-node n with child nodes n' and n'' , we have

$$\mathcal{E}_n(\mathcal{P}) = \bigcup_{(\mathcal{P}', \mathcal{P}'') \prec_n^{\min} \mathcal{P}} \{\Gamma' \cup \Gamma'' : \Gamma' \in \mathcal{E}_{n'}(\mathcal{P}') \text{ and } \Gamma'' \in \mathcal{E}_{n''}(\mathcal{P}'')\}.$$

Analogously to the cases of (R)- and (I)-nodes, it suffices to show that this is a disjoint union, i.e., for $i \in \{1, 2\}$, let $\Gamma'_i \in \mathcal{E}_{n'}(\mathcal{P}')$ and $\Gamma''_i \in \mathcal{E}_{n''}(\mathcal{P}'')$ with either $\Gamma'_1 \neq \Gamma'_2$ or $\Gamma''_1 \neq \Gamma''_2$. We have to show that then $\Gamma'_1 \cup \Gamma''_1 \neq \Gamma'_2 \cup \Gamma''_2$ holds. By symmetry, it suffices to consider the case $\Gamma'_1 \neq \Gamma'_2$. By definition of $\mathcal{E}_{n'}(\cdot)$ and $\Gamma(\cdot)$, the sets Γ'_1 and Γ'_2 coincide on the vertices in $\chi(n')$ (which is identical to $\chi(n)$ and $\chi(n'')$ by definition of (B) nodes). Hence, Γ'_1 and Γ'_2 differ on some edge e where at least one endpoint is contained in $\chi(T_{n'}) \setminus \chi(n')$. W.l.o.g., suppose that e is in Γ'_1 but not in Γ'_2 . By the connectedness condition of tree decompositions, e cannot be contained in Γ''_2 . Hence, also $\Gamma'_1 \cup \Gamma''_1$ and $\Gamma'_2 \cup \Gamma''_2$ differ on e , i.e., $e \in \Gamma'_1 \cup \Gamma''_1$ but $e \notin \Gamma'_2 \cup \Gamma''_2$.

A.9 Proof of Theorem 3

The theorem is proved by analogous arguments as in Theorem 1: We can compute $\pi_n(\mathcal{P})$ and, simultaneously, $\#_n(\mathcal{P})$ for every pcoloring \mathcal{P} at every node n in the tree decomposition T by a single bottom-up traversal of T and by applying the inductive definition of $\pi_n(\mathcal{P})$ and $\#_n(\mathcal{P})$ according to Definitions 11 and 12. Let $w = w^* + 1$. At every node n , there are at most $w!$ possible colorings \mathcal{P}_c . Moreover, for every \mathcal{P}_c (which is a partition of w vertices), there are at most 2^{w^2} possible values for \mathcal{P}_d . In total, we thus have $O(w! \cdot 2^{w^2})$ possible pcolorings $\mathcal{P} = (\mathcal{P}_c, \mathcal{P}_d)$ at each node n in the tree decomposition T .

Analogously to the proof of Theorem 1, $\pi_n(\mathcal{P})$ and $\#_n(\mathcal{P})$ can be computed in nested loops over pairs of pcolorings, s.t. the action performed for each pair fits into time which is polynomially bounded in w . In total, the effort required for processing each node n in T is bounded by $O(w! \cdot w! \cdot 2^{2w^2} \cdot p(w)) = O(2^k)$ with $k = 4 \cdot (w^*)^2$. The cost for computing the first solution and any further solution by a single top-down traversal of T clearly also fits into this bound. Thus, the desired time bound $O(2^k \cdot \|(G, H)\|)$ of Theorem 3 is proved.

B Enumeration

In this section, we provide a more detailed presentation of an enumeration algorithm which has been informally sketched in Section 4. The main algorithm is depicted in Figure 3. The functioning of this algorithm together with the definition of the helper-functions `reset(\cdot)`, `getChildren(\cdot , \cdot)`, and `nextChild(\cdot , \cdot)` will be discussed below in detail.

Our algorithm is supposed to be started after a bottom-up traversal of the tree-decomposition as discussed in Section 4. We assume that some of the information gathered during this bottom-up traversal is still available. In particular, we need the valid pcolorings of the nodes n together with the relation \prec_n^{\min} .

First, we observe that our enumeration algorithm `enumerateMinCuts` starts at the root node of \mathcal{T} . Note that this is exactly what we have to do, since the

```

Function getMinCut( $n, \mathcal{P}, isLast$ )
: node  $n \in T$ , pcoloring  $\mathcal{P}$  of  $n$ , boolean  $isLast$ 
		return: cut  $F \subseteq E$ , boolean  $isLast$ 
begin
		if  $n$  is (L) node then return  $(\Gamma(\mathcal{P}_c), isLast)$ ;
		if  $n$  is (B) node with children  $n', n''$  then
			 $(\mathcal{P}', \mathcal{P}'') = \text{getChildren}(n, \mathcal{P})$ ;
			 $(F', isLast) = \text{getMinCut}(n', \mathcal{P}', isLast)$ ;
			 $(F'', isLast) = \text{getMinCut}(n'', \mathcal{P}'', isLast)$ ;
			 $F = F' \cup F''$ ;
		if  $n$  is (I) node with child  $n'$  then
			 $\mathcal{P}' = \text{getChildren}(n, \mathcal{P})$ ;
			 $(F', isLast) = \text{getMinCut}(n', \mathcal{P}', isLast)$ ;
			 $F = F' \cup \Gamma(\mathcal{P}_c)$ ;
		if  $n$  is (R) node with child  $n'$  then
			 $\mathcal{P}' = \text{getChildren}(n, \mathcal{P})$ ;
			 $(F, isLast) = \text{getMinCut}(n', \mathcal{P}', isLast)$ ;
		endif
		if  $isLast$  then  $isLast = \text{nextChild}(n, \mathcal{P})$ ;
		return  $(F, isLast)$ ;
end

Program enumerateMinCuts
begin
		for each  $\mathcal{P} \in \mathbf{P}_{min}$  do
			reset $(\mathcal{P})$ ;
			repeat
				 $(F, isLast) = \text{getMinCut}(n_{root}, \mathcal{P}, \text{true})$ ;
				output  $F$ ;
			until  $isLast$ ;
		done
end

```

Fig. 3. Program `enumerateMinCuts`.

root is the only node where pcolorings \mathcal{P} which correspond to the minimum cuts of the given problem are identified (see e.g. Theorem 2). More precisely, our algorithm iterates over all $\mathcal{P} \in \mathbf{P}_{min}$ (recall that \mathbf{P}_{min} is the set of all valid pcolorings \mathcal{P} of n_{root} , for which there exists no other pcoloring \mathcal{P}' of n_{root} , such that $\sigma_{n_{root}}(\mathcal{P}') < \sigma_{n_{root}}(\mathcal{P})$). For each such $\mathcal{P} \in \mathbf{P}_{min}$, each call to the recursive function `getMinCut` $(n_{root}, \mathcal{P}, \text{true})$ yields one element from $\mathcal{E}_{n_{root}}(\mathcal{P})$ until the entire set $\mathcal{E}_{n_{root}}(\mathcal{P})$ has been enumerated. This situation is indicated by the flag $isLast$ which plays an important role in `getMinCut` (\cdot, \cdot, \cdot) . Roughly speaking, being a parameter in `getMinCut` $(n, \mathcal{P}, isLast)$, if $isLast$ is set to false, we will compute the same element of $\mathcal{E}_n(\mathcal{P})$ in the next call, otherwise we will compute a new element of $\mathcal{E}_n(\mathcal{P})$ in the next call. This toggling between same and new results for the subtree of n , is of importance to sequentialize the processing of all pcolorings $\mathcal{P}', \mathcal{P}''$ such that $\mathcal{P}' \prec_n^{min} \mathcal{P}$ (resp. $(\mathcal{P}', \mathcal{P}'') \prec_n^{min} \mathcal{P}$).

We next describe how these related pcolorings are treated in the algorithm and how the functions `reset` (\cdot) , `getChildren` (\cdot, \cdot) , and `nextChild` (\cdot, \cdot) manipulate these data structures. In principle, the data structure maintained for every

pcoloring \mathcal{P} of every node n contains an ordered list of pointers to the pcolorings \mathcal{P}' which fulfill $\mathcal{P}' \prec_n^{min} \mathcal{P}$. In case of a (B)-node n , there is an ordered list of pairs of pointers such that we can obtain pairs $(\mathcal{P}', \mathcal{P}'')$, which fulfill $(\mathcal{P}', \mathcal{P}'') \prec_n^{min} \mathcal{P}$. For each such list there will be a *mark*, which can be moved from pointer to pointer. The manipulation of the mark is implicit in function `nextChild(\cdot, \cdot)`.

- Function `getChildren(n, \mathcal{P})` first checks whether the pointer list for \mathcal{P} at n is already initialized. If not, it collects pointers (in an arbitrary but fixed order) to all \mathcal{P}' such that $\mathcal{P}' \prec_n^{min} \mathcal{P}$ (in case of an (I)- or (R)-node). For a (B)-node, pairs of pointers to all $(\mathcal{P}', \mathcal{P}'') \prec_n^{min} \mathcal{P}$ are collected in the same way. Recall that we have these pcolorings indeed available from the tree-decomposition bottom-up traversal as sketched in Section 4. Finally, the initialization step sets the mark to the first pointer (resp. to the first pair of pointers).
After this initialization, or in case the data structure has been already initialized earlier, `getChildren(n, \mathcal{P})` yields as its return value the pcoloring \mathcal{P}' , resp. the pair $(\mathcal{P}', \mathcal{P}'')$, to which the currently marked pointer refers.
- Function `nextChild(n, \mathcal{P})` just sets the mark to the next pointer, or, in case, the mark was already at the last pointer, it sets the mark to the first pointer. However, the return value depends on this distinction: if we have reset the mark to the first pointer, `nextChild(n, \mathcal{P})` returns `true`, otherwise it returns `false`.
- Function `reset(\cdot)` removes all these data structures in order to signalize function `getChildren(n, \mathcal{P})` that it has to initialize its list of pointers.

In the remainder of the section, we describe the pseudo-code of the function `getMinCut($n, \mathcal{P}, isLast$)` step-by-step.

- In case n is an (L) node, we return the induced cut of \mathcal{P}_c as a solution together with the parameter *isLast* unchanged. In case *isLast* is true, the calling procedure asks whether we can locally provide a new solution in the next iteration. However, at a leaf node there is only a single such solution, namely $\Gamma(\mathcal{P}_c)$. Thus we return `true` indicating that we have already reached the last possible solution which can be locally obtained. In case, *isLast* is false, we are not asked to provide a new solution for the next iteration. Here, we just pass `false` as a return value.
- In case n is a (B) node, we first pick up the current pair of pcolorings $(\mathcal{P}', \mathcal{P}'')$ for the child nodes of n via `getChildren(n, \mathcal{P})` and call `getMinCut` for these two pcolorings. Here, the role of *isLast* has to be understood as follows: If parameter *isLast* has been set to true, we are asked to provide a new solution in the next call. We first, pass this question to the left subtree and call `getMinCut($n', \mathcal{P}', true$)`. We get as return value of this call a partial solution F' for the left subtree and a boolean value (also called *isLast* for keeping the pseudo code concise).

In case this boolean value is `false`, we know that a new solution for the next call is already provided within this left subtree; so there is no need to change the solution in the right subtree for the next iteration and thus we

call `getMinCut(n'' , \mathcal{P}'' , false)`; as well there is no need to change the mark of the pointer list for \mathcal{P} itself (see the “*if isLast*” statement and the end of function `getMinCut`).

In case this boolean is `true`, we know that the left-subtree has run through all partial solutions which can be obtained there, and in the next call it will start with the very first solution again (all marks have been set to the first element of the respective pointer lists). Thus, we have to ask the right subtree, whether a new solution can be provided for the next iteration. Thus we call here `getMinCut(n'' , \mathcal{P}'' , true)`.

Now, in case the right subtree cannot deliver a new solution either for the next iteration, we have `true` as a return value (together with the current partial solution F'' for the right subtree). Hence, we have now to move the current mark via `nextChild(n , \mathcal{P})`. Otherwise, there is no need to change the mark since a distinct solution will be provided for the next iteration, thanks to the new partial solution which will be made available by the right subtree.

Finally, we just have to combine the two solutions F' , F'' to the induced cut of \mathcal{P}_c at n , i.e. $F = F' \cup F''$.

- In case n is an (I) node, we first pick up the current pcoloring \mathcal{P}' for the child node of n via `getChildren(n , \mathcal{P})`. We recursively call our function via `getMinCut(n' , \mathcal{P}' , isLast)`. The return value F' has to be augmented with the local induced cut; the boolean return value plays the same role as discussed in case of a (B) node. If *isLast* has been true as a parameter, we have to provide a new partial solution for the next iteration; thus in case this is already done in the subtree, there is no need to change to a new \mathcal{P}' for the next call. In this case, the return value *isLast* from `getMinCut(n' , \mathcal{P}' , isLast)` is `false`. Otherwise, we will need to call `nextChild(n , \mathcal{P})`; as we did for (B) nodes.
- The case where n is an (R) node is similar. Furthermore, we do not need to manipulate the return value F here.
- The final step is (as already outlined above) to check whether we have to move the mark for \mathcal{P} (i.e., whether we want to pick up a new \mathcal{P}' (a new pair $(\mathcal{P}', \mathcal{P}'')$) in the next call. This is the case only if *isLast* was true as a parameter, and each subrecursion has reported that it cannot provide a new subsolution by itself (indicated by the return values *isLast*). In such a case we move the mark via `nextChild(n , \mathcal{P})`. Recall that `nextChild(n , \mathcal{P})` reports whether it has reached the end of the list via its return value. In case the return value is `true`, we have to report the calling procedure that no new solution can be provided for the next iteration and return `true`, otherwise, we can return `false`, indicating that we did not reach the last subsolution yet. The other return value, F , carries the currently obtained subsolution as discussed above for the different cases of (B)-, (I)-, and (R)-nodes.