Parameterized Complexity of Weighted Multicut in Trees

Esther Galby[1]
esther.galby@cispa.de

Dániel Marx[1]
marx@cispa.de

Philipp Schepper[1]
philipp.schepper@cispa.de

Roohani Sharma[2]
rsharma@mpi-inf.mpg.de

Prafullkumar Tale[1]
prafullkumar.tale@cispa.de

[1] CISPA Helmholtz Center for Information Security, Germany
[2] Max Planck Institute for Informatics, SIC, Germany

Abstract

The EDGE MULTICUT problem is a classical cut problem where given an undirected graph G, a set of pairs of vertices $\mathcal{P}$, and a budget k, the goal is to determine if there is a set S of at most k edges such that for each $(s, t) \in \mathcal{P}$, $G - S$ has no path from s to t. EDGE MULTICUT has been relatively recently shown to be fixed-parameter tractable (FPT), parameterized by k, by Marx and Razgon [SICOMP 2014], and independently by Bousquet et al. [SICOMP 2018]. In the weighted version of the problem, called WEIGHTED EDGE MULTICUT one is additionally given a weight function $\mathtt{wt} : E(G) \to \mathbb{N}$ and a weight bound $\mathbf{w}$, and the goal is to determine if there is a solution of size at most k and weight at most $\mathbf{w}$. Both the FPT algorithms for EDGE MULTICUT by Marx et al. and Bousquet et al. fail to generalize to the weighted setting. In fact, the weighted problem is non-trivial even on trees and determining whether WEIGHTED EDGE MULTICUT on trees is FPT was explicitly posed as an open problem by Bousquet et al. [STACS 2009]. In this article, we answer this question positively by designing an algorithm which uses a very recent result by Kim et al. [STOC 2022] about directed flow augmentation as subroutine.

We also study a variant of this problem where there is no bound on the size of the solution, but the parameter is a structural property of the input, for example, the number of leaves of the tree. We strengthen our results by stating them for the more general vertex deletion version.

1 Introduction

EDGE MULTICUT is a generalization of the classical (s, t)-CUT problem where given a graph G, a set of terminal pairs $\mathcal{P} = \{(s_1, t_1), \ldots, (s_p, t_p)\}$, and an integer k, the goal is to determine if there exists a set of at most k edges whose deletion disconnects s_i from t_i, for each $i \in [p]$. Such a set is called a $\mathcal{P}$-*multicut* in G. The case $p = 1$ corresponds to the classical (s, t)-CUT problem. EDGE MULTICUT is polynomial time solvable for $p \leq 2$ [23] and is NP-hard even for $p = 3$ [8]. From the parameterized complexity point of view, it was a long-standing open question to determine if the problem is fixed-parameter tractable (FPT) parameterized by the solution size. This question was resolved independently by Marx and Razgon [22] and Bousquet et al. [2], proving that the problem is FPT. Both algorithms extensively use the notion of important separators, a technique introduced earlier by Marx [21]. Bousquet et al. [2] additionally use several problem-specific observations and arguments about the structure of multicut instances, while Marx and Razgon [22] formulated the technique of random sampling of important separators, which found further applications for many other problems [4, 5, 6, 18, 19, 20].

Weighted Multicut. One drawback of the algorithms using important separators is that they are essentially based on a replacement argument: if a subset X of the solution satisfies

some property, then this technique allows us to find a set X' such that X can be replaced with X', thereby making progress towards fully identifying a solution. This local replacement argument inherently fails if the overall solution is also required to satisfy additional properties, such as minimizing the overall weight, since replacing X with X' may violate these additional constraints. Thus, the ideas from the algorithms of Marx and Razgon [22] and Bousquet et al. [2] fail to generalize to the edge deletion version of WEIGHTED MULTICUT (wMC) where we are, additionally, given a weight function $\mathtt{wt} : E(G) \to \mathbb{N}$ and an integer $\mathbf{w}$, and the goal is to determine if there exists a $\mathcal{P}$-multicut in G of size at most k and weight at most $\mathbf{w}$.

(Weighted) Multicut on Trees. EDGE MULTICUT remains NP-hard on trees [11] but can be solved in $\mathcal{O}(2^k \cdot n)$-time, where n is the number of vertices in the input tree, using an easy branching algorithm [12]: for the "deepest" (s_i, t_i)-path branch on the deletion of the two edges on this path which are incident to the lowest common ancestor of s_i and t_i. A series of work shows improvement over this simple running time [3, 14], and also the problem admits a polynomial kernel [1, 3]. Since the algorithmic approaches for EDGE MULTICUT on trees are based on greedily finding partial solutions, they do not generalize to the weighted setting. In fact, the question whether wMC on trees is FPT (parameterized by the solution size), was explicitly posed as an open problem by Bousquet et al. [1]. In this article, we answer this question in the positive.

Flow Augmentation. As mentioned earlier, most of the available techniques used to design FPT algorithms, especially for cut problems, do not work in the weighted setting. Kim et al. [15] recently developed the technique of flow augmentation in directed graphs. This technique offers a new perspective to design FPT algorithms for cut problems and positively settles the parameterized complexity of some long standing open problems, such as WEIGHTED (s, t)-CUT, WEIGHTED DIRECTED FEEDBACK VERTEX SET and WEIGHTED DIGRAPH PAIR CUT.

Our main goal is to use this technique for the underlying core difficulty in wMC on trees. More precisely, we do not use the directed flow augmentation technique as such but we crucially use the FPT algorithm for WEIGHTED DIGRAPH PAIR CUT (wDPC) which is one important example of the use of this technique. The wDPC problem is defined as follows [15, 17]: given a *directed* graph G, a source vertex $\mathbf{r} \in V(G)$, terminal pairs $\mathcal{P} = \{(s_1, t_1), \ldots, (s_p, t_p)\}$, a weight function $\mathtt{wt} : E(G) \to \mathbb{N}$, a positive integer k, the goal is to determine if there exists a set S of at most k arcs of G such that $\mathtt{wt}(S)$ is minimum[1], and for each $i \in [p]$, if $G - S$ has a path from $\mathbf{r}$ to s_i, then $G - S$ has no path from $\mathbf{r}$ to t_i. Such a set is called a $\mathcal{P}$-*dpc* with respect to $\mathbf{r}$ in G. Kim et al. [15, Section 6.1ff] showed that wDPC can be solved in randomized $2^{\mathcal{O}(k^4)} \cdot n^{\mathcal{O}(1)}$-time. The randomized running time of this algorithm is an artifact of the use of the directed flow augmentation procedure which is randomized. Apart from this step, all the other steps of the algorithm are deterministic. Our basic observation is that the algorithm for wDPC can be used to solve a non-trivial base case of wMC in trees: if there is a vertex $\mathbf{r} \in V(T)$ such that all the terminal pair paths of $\mathcal{P}$ pass through $\mathbf{r}$, then S is a $\mathcal{P}$-multicut of T if and only if for all $(s, t) \in \mathcal{P}$, S intersects the $(\mathbf{r}, s)$-path or the $(\mathbf{r}, t)$-path. This is equivalent to saying that S is a $\mathcal{P}$-dpc for T (in wDPC we interpret each edge of T to be directed away from $\mathbf{r}$).[2]

Edge Deletion vs. Vertex Deletion. In the weighted setting, the edge deletion version of wMC (on trees) reduces to its vertex deletion version (on trees), by subdividing each edge and assigning the weight of the original edge to the newly added vertex corresponding to the

---

[1]Though the formal description of the problem in [15] asks for a solution S with $\mathtt{wt}(S) \leq \mathbf{w}$, the authors remark that the algorithm in fact finds a minimum weight solution.

[2]When dealing with undirected graphs, the flow augmentation restricted to undirected graphs given by Kim et al. [16] may suffice to solve wDPC on undirected graphs. As this problem is not mentioned explicitly in [16], we stick to the directed setting.

edge, and by setting the weights of the original vertices to ∞ (or larger than the weight budget parameter). Note that such a reduction does not work in the unweighted setting as the vertex deletion version of MULTICUT in trees is polynomial time solvable [7].

Main Result. From now on we only study the vertex deletion version of WMC on trees which, as mentioned above, is more general than the edge deletion version. It is formally defined below.

WEIGHTED MULTICUT ON TREE (WMC-TREE)
Input: A tree T, a collection of terminal pairs $\mathcal{P} \subseteq V(T) \times V(T)$, a vertex weight function $\mathtt{wt} : V(T) \to \mathbb{N}$, and positive integers $\mathbf{w}$ and k.
Question: Does there exist $S \subseteq V(T)$ such that $|S| \leq k$, $\mathtt{wt}(S) \leq \mathbf{w}$, and S intersects the unique (s, t)-path in T, for each $(s, t) \in \mathcal{P}$?

We set $\mathtt{wt}(S) = \sum_{v \in S} \mathtt{wt}(v)$ for the ease of notation. We use the FPT algorithm for wDPC (restricted to trees) [15, Section 6.1ff] as a subroutine to prove our main result, namely that WMC-TREE is FPT.

Theorem 1.1. WMC-TREE *can be solved in randomized* $2^{\mathcal{O}(k^4)} \cdot n^{\mathcal{O}(1)}$ *time.*

Structural Parameterizations. In scenarios where the size of the solution is large, it might be desired to drop the constraint on the size of the solution altogether, and seek to parameterize the problem with some structural parameter of the input. In this setting, we first consider the problem parameterized by the number of leaves of the tree and then extend this result to a more general parameter that takes into account the number of requests (terminal pair paths) passing through a vertex. Technically, we solve a different problem in this setting, where we only have a uni-objective function seeking to minimize the weight of the solution (in contrast to the bi-objective function in the case of WMC-TREE). This problem is formally defined below.

UNCONSTRAINED WEIGHTED MULTICUT ON TREE (UWMC-TREE)
Input: A tree T, a collection of terminal pairs $\mathcal{P} \subseteq V(T) \times V(T)$, a vertex weight function $\mathtt{wt} : V(T) \to \mathbb{N}$ and a positive integer $\mathbf{w}$.
Question: Does there exist $S \subseteq V(T)$ such that $\mathtt{wt}(S) \leq \mathbf{w}$ and S intersects the unique (s, t)-path in T, for each $(s, t) \in \mathcal{P}$?

UWMC is another generalization of the vertex deletion variant of MULTICUT. The former problem has been studied on trees in the parameterized complexity setting with respect to certain structural parameters. In particular, Guo et al. [13, Theorem 9] showed that UWMC-TREE is FPT when the parameter is the maximum number of (s, t)-paths that pass through any vertex of the input. We call this parameter the *request degree* d of an instance. Guo et al. [13] gave an algorithm for UWMC-TREE that runs in time $\mathcal{O}(3^d \cdot n)$.

We first study UWMC-TREE when the parameter is the number of leaves of the tree. The problem is polynomial time solvable on paths (Lemma 4.1) but becomes NP-hard on (general) trees. Thus, the number of leaves appears to be a natural parameter which could explain the contrast between the above two results. Formally, we prove the following theorem.

Theorem 1.2. UWMC-TREE *can be solved in* $2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n^{\mathcal{O}(1)}$ *time, where* ℓ *is the number of leaves in the input tree.*

At the core of the algorithm for Theorem 1.2, we again solve instances of wDPC on trees, but, in this case, these instances have a special structure: they are subdivided stars (i.e. trees with at most one vertex of degree at least 3). We show that these instances do not require the use of the flow augmentation technique. In fact, these instances correspond to the arcless instances of wDPC in [15, Section 6.2.2] defined roughly as follows: the input graph comprises of two designated vertices s, t with internally vertex-disjoint paths from s to t, and the solution picks exactly one arc from each of these internally vertex-disjoint paths. Since the arcless instances

can be solved faster than the general instances of wDPC and do not require the usage of the flow augmentation technique [15, Lemma 6.12], the algorithm for UWMC-Tree is deterministic and has a better running time.

As a final result, we use the algorithm of Theorem 1.2 as a subroutine to give an FPT algorithm for UWMC-Tree that generalizes the result of Guo et al. [13, Theorem 9] and Theorem 1.2. To do so, we define a new parameter that comprises both the request degree and the number of leaves of the input instance. An instance $(T, \mathcal{P}, \mathtt{wt}, \mathbf{w})$ is (d, q)-*light* if the following hold. Let Y be the set of vertices through which at most d terminal pair paths of $\mathcal{P}$ pass. Such vertices are called d-*light vertices*. Then for each connected component C of $T - Y$, the number of leaves of $T[N[C]]$ must be at most q (see Figure 6 for an illustration of the definition). We observe in Section 5 that it is crucial to consider the *neighborhood* of the component, as the problem is otherwise already NP-hard for $d = 3$ and $q = 2$. We design a dynamic programming algorithm that stores partial solutions for every d-light vertex using the algorithm of Theorem 1.2 as a subroutine to solve the problem on (d, q)-light instances.

Theorem 1.3. UWMC-Tree *can be solved in* $3^d \cdot 2^{dq} \cdot 2^{\mathcal{O}(q^2 \log q)} \cdot n^{\mathcal{O}(1)}$ *time on* (d, q)-*light instances.*

Observe that an instance with a tree on ℓ leaves is a $(0, \ell)$-light instance, and an instance with the request degree at most d is a $(d, 0)$-light instance. Thus, Theorem 1.3 implies Theorem 1.2 and Theorem 9 in [13], up to the polynomial factors in the running time.

Our Methods. Our algorithms for Theorems 1.1 and 1.2, are crucially based on the observation mentioned earlier: if every terminal path goes through a root $\mathbf{r}$, then the problem reduces to wDPC. In the vertex deletion version, the *vertex* $\mathcal{P}$-multicut in a tree can be found using the algorithm for wDPC, by assigning the weight of a vertex to the unique edge connecting it to its parent in T. The general idea for both our algorithms is to design a branching algorithm that effectively solves instances of the above-mentioned type to reduce the measure in each branch. Let T be a rooted tree. The goal is to identify two vertices $x, y \in V(T)$ where x is a descendant of y, and branch on the possibility of a hypothetical solution intersecting the (y, x)-path. If the solution does not intersect the (y, x)-path, then contracting the edges of the (y, x)-path and making the resulting vertex undeletable, is a safe operation. If the solution intersects the (y, x)-path, then for each vertex v on the (y, x)-path, we increase the weight of v by adding to it the minimum weight of a solution in $T_v - \{v\}$ (where T_v is the subtree of T rooted at V), and then forget about the terminal pair paths in $T_v - \{v\}$. To update the weight of v, one therefore needs to find a minimum weight solution in $T_v - \{v\}$. For this reason, we choose the vertices x, y so that the instance restricted to $T_v - \{v\}$ can be solved using the algorithm for wDPC.

If x, y are vertices of degree at least 3 (branching vertices) in T, then contracting the (y, x)-path decreases the number of branching vertices in the resulting instance. This choice of x, y allows to design a branching algorithm where the measure is the number of branching vertices, and thus the number of leaves (Theorem 1.2). If x, y are vertices of a minimum-size (unweighted) $\mathcal{P}$-multicut (which can be found in polynomial time), then contracting the (y, x)-path decreases the size of a $\mathcal{P}$-multicut in the resulting instance. This choice of x, y allows the design of a branching algorithm parameterized by the solution size (Theorem 1.1). Additionally, if we choose x to be the furthest branching vertex in T (resp. furthest vertex of X) from the root and y to be its unique closest ancestor that is a branching vertex (resp. in X), then for each vertex v on the (y, x)-path, the instance restricted on $T_v - \{v\}$ can indeed be solved using the algorithm for wDPC.

Organization. In Section 2 we define some basic notation. In Section 3 we prove Theorem 1.1, in Section 4 we prove Theorem 1.2, and in Section 5 we prove Theorem 1.3. We finally conclude in Section 6.
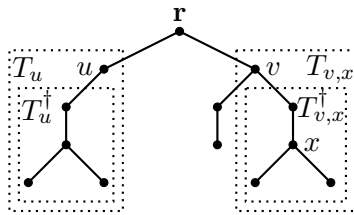
Figure 1: Tree rooted at $\mathbf{r}$ and an illustration of $T_u, T_u^\dagger, T_{v,x}, T_{v,x}^\dagger$.

2 Notation and Preliminaries

For a positive integer n, we denote the set $\{1, 2, \ldots, n\}$ by $[n]$. We use $\mathbb{N}$ to denote the set of all non-negative integers. Given a function $f : X \to Z$ and $Y \subseteq X$, $f|_Y$ denotes the function f restricted to Y.

Graph Theory. For a (di-)graph G, $V(G)$ and $E(G)$ denote the set of vertices and edges (arcs) of G, respectively. For an undirected graph G and any $v \in V(G)$, $N_G(v) = \{u : (u, v) \in E(G)\}$ denotes the (open) neighbourhood of v, and $N_G[v] = N_G(v) \cup \{v\}$ denotes the closed neighbourhood of v. The degree of v is $|N_G(v)|$. For a digraph G and $v \in V(G)$, the *in-degree* of v is the number of vertices u such that $(u, v) \in E(G)$. When the graph G is clear from the context we omit subscript G. For any $u, v \in V(G)$, a (u, v)-path in G denotes a path from u to v in G. For any $S \subseteq V(G)$, $G[S]$ denotes the graph G induced on S. We say $G' \subseteq G$ if G' is a subgraph of G. For any further notation from basic graph theory, we refer the reader to [9].

Contraction. For any edge $e = (u, v)$ of G, G/e represents the graph G obtained after contracting the edge e, where the contraction of e is defined as follows: delete u, v from G and add a new vertex say x_{uv} that is adjacent to the all the vertices in $N[u] \cup N[v] \setminus \{u, v\}$, that is all vertices that were either adjacent to u or v or both. We sometime also say that the edge e is contracted onto the vertex x_{uv}. For any $F \subseteq E(G)$, G/F denotes the graph obtained by contracting the edges of F (one after the other in no specified order). Formally speaking G/F corresponds to a unique map $\psi : V(G) \to V(G \setminus F)$, such that for each $u \in V(G/F), G[\psi^{-1}(u)]$ is connected. Let $\mathcal{P} \subseteq V(G) \times V(G)$ be a set of pairs of vertices, then $\mathcal{P}/F$ is obtained from $\mathcal{P}$ by replacing each $(u, v) \in \mathcal{P}$ by $(\psi(u), \psi(v))$. Given $\mathcal{P} \subseteq V(G) \times V(G)$ and an induced subgraph G' of G, $\mathcal{P}|_{G'} \subseteq \mathcal{P}$ such that if $(u, v) \in \mathcal{P}$ then $(u, v) \in \mathcal{P}|_{G'}$ if and only if $u, v \in V(G')$.

Terminal Pairs and Terminal Pair Paths. Recall that the instances of all our problems contain a set $\mathcal{P} \subseteq V(T) \times V(T)$ where T is a tree. We interchangeably refer to a pair $(s, t) \in \mathcal{P}$ as the terminal pair (s, t) and as the unique (s, t)-path between in T.

Trees. A *tree* T is a connected acyclic graph. A *subdivided star* is a tree with at most one vertex of degree at least 3 (in other words, it is a tree obtained by repeatedly sub-dividing the edges of a star graph). For any $x, y \in V(T)$, $P_{x,y}$ denotes the unique (x, y)-path in T. Note that the vertices of such a path $P_{x,y}$ are ordered starting from x thus they have a natural order defined on them. Let T be a tree rooted at a vertex $\mathbf{r}$. A vertex $u \in V(T)$ is called a *furthest* vertex from v if $\mathsf{dist}_T(u, v) = \max_{x \in V(T)}\{\mathsf{dist}_T(x, v)\}$. Here, $\mathsf{dist}_T(u, v)$ denotes the length of the shortest (u, v)-path in T. Similarly, $u \in V(T) \setminus \{\mathbf{r}\}$ is closest to v, if $0 < \mathsf{dist}_T(u, v) = \min_{x \in V(T)}\{\mathsf{dist}_T(x, v)\}$. We say that a vertex is *furthest* (resp. *closest*) if it is furthest (resp. closest) from $\mathbf{r}$. The sets $V_{\geq 3}(T)$ and $V_{=1}(T)$ denote the set of vertices of degree at least 3, and of degree equal to 1, respectively. The set $V_{\geq 3}(T)$ is also called the set of *branching vertices* of T and the set $V_{=1}(T)$ is called the set of *leaves* of T. Note that $|V_{\geq 3}(T)| \leq |V_{=1}(T)| - 1$.

The following notation (see Figure 1) comes handy while describing the algorithms on a tree T with root $\mathbf{r}$. Given $u, v \in V(T)$, u is a *descendant* of v in T, if v lies on the unique $(\mathbf{r}, u)$ path in T and u is called an *ancestor* of v if u lies on the unique $(\mathbf{r}, v)$-path in T (u could be equal to v). We denote by T_u the subtree of T rooted at u and $T_u^{\dagger} = T_u \setminus \{u\}$. For any descendant x of u, the tree denoted by $T_{u,x}$ is defined as follows. Let $\{v_1, \ldots, v_p\}$ be the children of u in T and say $x \in V(T_{v_i})$. Then $T_{u,x} = T_u \setminus (\cup_{j \in [p] \setminus \{i\}} T_{v_j})$. Observe that $T_{u,x}$ is connected. Also define $T_{u,x}^{\dagger} = T_{u,x} \setminus \{u\}$.

Let $X \subseteq V(T)$, then the `lca-closure` of X is the set X' obtained from X by repeatedly adding, for each pair of vertices $u, v \in X'$, the least common ancestor w to X', that is the vertex $w \in V(T)$ furthest from $\mathbf{r}$ such that $u, v \in V(T_w)$. Note that $|X'| \leq 2|X|$ (because $|V_{\geq 3}(T)| < |V_{=1}|(T)|$). We say that a set X is *closed under taking* `lca` if for every pair of vertices of X, their least common ancestor is in X. Whenever we talk about a rooted (undirected) tree T in a directed setting, we refer to the tree T where each vertex except the root has in-degree exactly one.
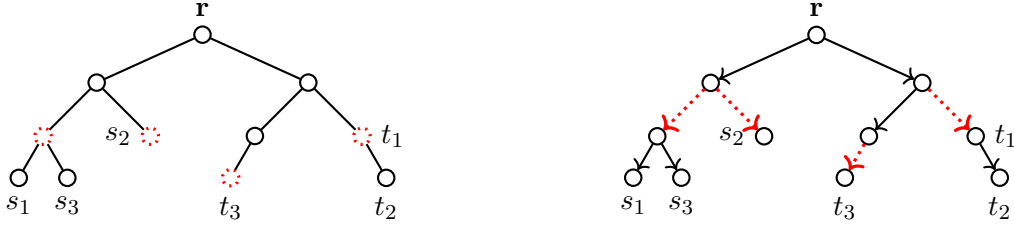
Parameterized Complexity. The input of a parameterized problem comprises of an instance I, which is an input of the classical instance of the problem, and an integer k, which is called the parameter. A problem Π is said to be *fixed-parameter tractable* or FPT if given an instance (I, k) of Π, we can decide whether (I, k) is a YES instance of Π in time $f(k) \cdot |I|^{\mathcal{O}(1)}$. Here, $f(\cdot)$ is some computable function whose value depends only on k. We say that two instances, (I, k) and (I', k'), of a parameterized problem Π are *equivalent* if $(I, k) \in \Pi$ if and only if $(I', k') \in \Pi$. For more details on parameterized algorithms, and in particular parameterized branching algorithms, we refer the reader to the book by Cygan et al. [7].

3 wMC-Tree Parameterized by the Solution Size

In this section, we prove Theorem 1.1 by designing a branching algorithm. In order to reduce the measure of a given instance, our branching algorithm requires a solution for the instances where every terminal pair path passes through a single vertex. Let $\mathcal{I} = (T, \mathcal{P}, \mathbf{r}, \mathtt{wt}, k)$ be an instance such that all the terminal pair paths of $\mathcal{P}$ pass through $\mathbf{r}$, and $\mathtt{wt} : V(T) \to \mathbb{N}$ is a vertex weight function. Let $\overrightarrow{T}$ be the directed tree obtained by orienting the edges of T so that all the vertices, except for $\mathbf{r}$, have in-degree exactly one, while $\mathbf{r}$ has in-degree zero. In other words, the oriented tree $\overrightarrow{T}$ is an out-tree rooted at $\mathbf{r}$. We define an edge weight function $\mathtt{wt}' : E(\overrightarrow{T}) \to \mathbb{N}$ such that for every arc $e = (u, v) \in E(\overrightarrow{T})$, $\mathtt{wt}'(e) = \mathtt{wt}(v)$. Then it can be easily seen that $Z \subseteq E(\overrightarrow{T})$ is a $\mathcal{P}$-dpc in T with $\mathtt{wt}'(Z) = \mathbf{w}$ if and only if $S = \{v : (u, v) \in Z\} \subseteq V(T) \setminus \{\mathbf{r}\}$ (that is, S is obtained from Z by picking the heads of all the arcs in Z) is a $\mathcal{P}$-multicut in T with $\mathtt{wt}(S) = \mathbf{w}$ (see Figure 2). Let $\mathcal{A}_{\mathsf{dpc}}$ be the algorithm that takes as input an instance $\mathcal{I}$ as above, constructs the edge-weight function $\mathtt{wt}'$ and uses the wDPC algorithm of Kim et al. [15, Section 6.1ff] to solve the instance $(\overrightarrow{T}, \mathcal{P}, \mathbf{r}, \mathtt{wt}', k)$. This runs in randomized $2^{\mathcal{O}(k^4)} \cdot n^{\mathcal{O}(1)}$-time [3]. Therefore, $\mathcal{A}_{\mathsf{dpc}}$ outputs the minimum *weight* of a solution of $\mathcal{I}$ if it exists, and ∞ otherwise. In particular, if $\mathcal{P} = \emptyset$ then $\mathcal{A}_{\mathsf{dpc}}$ outputs 0.

Branching Algorithm. Let $(T, \mathcal{P}, \mathtt{wt}, \mathbf{w}, k)$ be an instance of wMC-Tree. Fix an arbitrary vertex $\mathbf{r} \in V(T)$ to be the root of T. We begin by finding a set $X \subseteq V(T)$ which is a $\mathcal{P}$-multicut in T and is closed under taking `lca` (least common ancestor). To find X, we first compute a *unweighted* $\mathcal{P}$-multicut $X_{\mathrm{opt}} \subseteq V(T)$ in T of minimum size. The set X_{opt} can be found in polynomial time (folklore) by the following greedy algorithm. Initialize $X_{\mathrm{opt}} = \emptyset, T' = T$, and $\mathcal{P}' = \mathcal{P}$. Let $v \in V(T')$ be a furthest vertex from $\mathbf{r}$ such that there exists $(s, t) \in \mathcal{P}'$

---

[3]The dependency in k is not explicit in [15] but can be easily deduced.

(a) The instance for wMC-Tree with a marked solution.

(b) The instance for wDPC where the corresponding solution is marked.

Figure 2: One-to-one correspondence between the solutions of wMC-Tree and the solutions of wDPC when all the paths of $\mathcal{P}$ pass through $\mathbf{r}$.

with $s, t \in V(T_v)$. By the choice of v, the (s,t)-path (and every terminal pair path in $\mathcal{P}|_{T_v}$) passes through v. It is easy to see that there is a minimum-size $\mathcal{P}'$-multicut containing v. Set $X_{\text{opt}} = X_{\text{opt}} \cup \{v\}$, $\mathcal{P}' = \mathcal{P}' \setminus \mathcal{P}|_{T_v}$, $T' = T' \setminus T_v$, and repeat the procedure until $\mathcal{P}' = \emptyset$. At the end of the procedure, X_{opt} is a minimum-size $\mathcal{P}$-multicut in T. If $|X_{\text{opt}}| > k$, report No. Otherwise, let X be the `lca-closure` of X_{opt} in T. Hence, $|X| \leq 2k$.

A notable property of a $\mathcal{P}$-multicut X closed under taking `lca` is that for any $x \in X$, if $y \in X$ is the unique closest ancestor of x in T, then for each $v \in V(P_{y,x}) \setminus \{y\}$, all the terminal pair paths of $\mathcal{P}|_{T_v}$ either pass through x, or are contained in T_x. Indeed, if $T_v \setminus T_x$ contains a path of $\mathcal{P}$, then any $\mathcal{P}$-multicut intersects $V(T_v \setminus T_x)$. Then there exists a vertex $y' \in X$ such that $y' \neq x$ lies on $P_{v,x} \subseteq P_{y,x}^{\dagger}$, contradicting the choice of y.

We design a branching algorithm whose input is $\mathcal{I} = (T, \mathcal{P}, \mathtt{wt}, \mathbf{w}, k, X)$ where X is $\mathcal{P}$-multicut $X \subseteq V(T)$ closed under taking `lca`, and where the measure of an instance $\mathcal{I}$ is defined as $\mu(\mathcal{I}) = |X|$. Note that, as mentioned above, $\mu(\mathcal{I}) \leq 2k$. The base case of the branching algorithm occurs in the following scenarios.

1. If $\mu(\mathcal{I}) = 0$, then $\emptyset$ is a solution of $\mathcal{I}$. Return Yes iff $k \geq 0$ and $\mathbf{w} \geq 0$.

2. If $\mu(\mathcal{I}) = 1$, let $X = \{x\}$. In this case, since all the paths of $\mathcal{P}$ pass through x, return Yes if and only if the $\mathcal{A}_{\mathsf{dpc}}(T, \mathcal{P}, x, \mathtt{wt}, k) \leq \mathbf{w}$.

3. If $k < 0$, or $k \leq 0$ and $\mathcal{P} \neq \emptyset$, then return No.

If $\mu(\mathcal{I}) \geq 2$ (that is, $|X| \geq 2$), then let $x \in X$ be a furthest vertex from $\mathbf{r}$ and let $y \in X$ be its unique closest ancestor. We branch in the following two cases.

Case 1. *There exists a solution of $\mathcal{I}$ that does not intersect $V(P_{y,x})$.* In this case, we return the instance $\mathcal{I}_1 = (T_1, \mathcal{P}_1, \mathtt{wt}_1, \mathbf{w}, k, X_1)$ where $T_1 = T/E(P_{y,x})$, $\mathcal{P}_1 = \mathcal{P}/E(P_{y,x})$. Let the vertex onto which the edges of $P_{y,x}$ are contracted be y°. Then $\mathtt{wt}_1(y^{\circ}) = \mathbf{w} + 1$ and, for each $v \in V(T_1) \setminus \{y^{\circ}\}$, $\mathtt{wt}_1(v) = \mathtt{wt}(v)$. Observe that $(X \setminus \{x, y\}) \cup \{y^{\circ}\}$ is a $\mathcal{P}_1$-multicut in T_1 and is closed under taking `lca` and thus, we may set $X_1 = (X \setminus \{x, y\}) \cup \{y^{\circ}\}$. Clearly, $\mu(\mathcal{I}_1) < \mu(\mathcal{I})$ and $\mathcal{I}_1$ can be constructed in polynomial time.

Case 2. *There exists a solution of $\mathcal{I}$ that intersects $V(P_{y,x})$.* In this case, the idea is the following: for each vertex v on the $P_{y,x}$ path, we increase the weight of v by the weight of the solution in the tree $T_{v,x}^{\dagger}$ (the tree strictly below v). To do so, the size of a solution in the tree $T_{v,x}^{\dagger}$ is first guessed. Once the weights are updated, we can forget the terminal pairs contained in the tree $T_{y,x}^{\dagger}$ and just remember that the solution picks a vertex from $P_{y,x}$. This is formalized below.

Let S be a solution which intersects $V(P_{y,x})$ and let $z \in V(P_{y,x})$ be the vertex in S closest to y. Then we further branch into $k+1$ branches where each branch corresponds to the guess on

(a) Case 1. The marked edges are contracted onto the undeletable y°.

(b) Case 3. $T_x^\dagger$ is deleted (dotted part). The weight of the filled vertices includes the weight of the solution from below.

Figure 3: The branches of the algorithm for Theorem 1.1 with $(s_i, t_i) \in \mathcal{P}$.

$|S \cap T_{z,x}^\dagger|$. More precisely, for every $i \in \{0\} \cup [k]$, we create the instance $\mathcal{I}_{2,i} = (T_2, \mathcal{P}_2, \mathtt{wt}_{2,i}, \mathbf{w}, k - i, X_2)$ where $T_2 = T \setminus T_x^\dagger$, $\mathcal{P}_2 = \mathcal{P}|_{T_2} \setminus (V(T_{y,x}^\dagger) \times V(T_{y,x}^\dagger)) \cup \{(y, x)\}$ and $\mathtt{wt}_{2,i}$ is defined below (see Figure 3).

$$\mathtt{wt}_{2,i}(v) = \begin{cases} \mathtt{wt}(v) + \mathcal{A}_{\mathsf{dpc}}(T_{v,x}^\dagger, \mathcal{P}|_{T_{v,x}^\dagger}, x, \mathtt{wt}|_{V(T_{v,x}^\dagger)}, i) & \text{if } v \in V(P_{y,x}) \\ \mathtt{wt}(v) & \text{otherwise.} \end{cases}$$

Observe that the set $X \setminus \{x\}$ is a $\mathcal{P}_2$-multicut in T_2 with $y \in X \setminus \{x\}$. The only paths that might not be cut are the ones in $\mathcal{P}|_{T_{y,x}^\dagger}$ as they pass through x, but they are not contained in $\mathcal{P}_2$ by definition. Also $X \setminus \{x\}$ is closed under taking $\mathtt{lca}$ in T_2, thus we may set $X_2 = X \setminus \{x\}$. Clearly, $\mu(\mathcal{I}_{2,i}) < \mu(\mathcal{I})$ for each $i \in \{0\} \cup [k]$.

Lemma 3.1. $\mathcal{I}$ is a YES-instance if and only if at least one of $\mathcal{I}_1, \mathcal{I}_{2,0}, \ldots, \mathcal{I}_{2,k}$ is a YES-instance.

Proof. ($\Rightarrow$) Assume that $\mathcal{I}$ is a YES-instance and let S be a minimal solution of $\mathcal{I}$. Suppose first that $S \cap V(P_{y,x}) = \emptyset$ and consider a path $P_{s,t}$ of $\mathcal{P}_1$. Then $(s, t) \neq (y^\circ, y^\circ)$ for otherwise, S would not intersect the path in $\mathcal{P}$ corresponding to the pair $(s, t) \in \mathcal{P}_1$. If $y^\circ \notin \{s, t\}$ then $(s, t) \in \mathcal{P}$ and so, S intersects the path $P_{s,t}$. Otherwise, assume, without loss of generality, that $s = y^\circ$ and let $(z, t) \in \mathcal{P}$ where $z \in V(P_{y,x})$, be the terminal pair in $\mathcal{P}$ corresponding to (s, t). Then, since $P_{z,t}$ is intersected by $S \setminus V(P_{y,x})$, $P_{s,t}$ is also intersected by $S \setminus \{y^\circ\}$. Thus, S is a solution for $\mathcal{I}_1$.

Suppose next that $S \cap V(P_{y,x}) \neq \emptyset$ and let $z \in V(P_{y,x})$ be the vertex in S closest to y. Observe that since X is a $\mathcal{P}$-multicut in T and $x \in X$ is a furthest vertex in T from $\mathbf{r}$, every path of $\mathcal{P}$ contained in T_x passes through x. Similarly, if $z \neq x$ then, from the choice of x and y, each terminal pair path contained in $T_{z,x}^\dagger$ passes through x: indeed, if there exists a terminal pair path contained in $T_{z,x}^\dagger \setminus T_x$, then it is not intersected by X, a contradiction to the fact that X is a $\mathcal{P}$-multicut. Let $S^* = S \cap T_{z,x}^\dagger$ and let $i = |S^*|$. Note that if $z = x$ then $\mathcal{P}_{T_{x,x}^\dagger} = \emptyset$ by the above, and thus, $S^* = \emptyset$ by minimality of S. Since S^* is a $\mathcal{P}|_{T_{z,x}^\dagger}$-multicut, it follows that $\mathtt{wt}(S^*) \geq \mathcal{A}_{\mathsf{dpc}}(T_{z,x}^\dagger, \mathcal{P}|_{T_{z,x}^\dagger}, x, \mathtt{wt}|_{T_{z,x}^\dagger}, i)$. Now let $S' = S \setminus S^*$. Note that $z \in S'$; in fact, $S' \cap V(P_{y,x}) = \{z\}$ by the choice of z. We claim that S' is a solution for $\mathcal{I}_{2,i}$. Clearly, $|S'| = |S| - |S^*| \leq k - i$. Furthermore, $\mathtt{wt}_{2,i}(S') = \mathtt{wt}(S) - \mathtt{wt}(S^*) - \mathtt{wt}(z) + \mathtt{wt}_{2,i}(z)$ and since $z \in V(P_{y,x})$, $\mathtt{wt}_{2,i}(z) \leq \mathtt{wt}(z) + \mathtt{wt}(S^*)$. Thus, $\mathtt{wt}_{2,i}(S') \leq \mathtt{wt}(S) \leq \mathbf{w}$. We now show that S' is a $\mathcal{P}_2$-multicut. Consider a path $P_{s,t}$ of $\mathcal{P}_2$. Since by construction, $\mathcal{P}_2 \cap (V(T_{y,x}^\dagger) \times V(T_{y,x}^\dagger)) = \emptyset$, at most one of s and t belongs to $V(T_{y,x}^\dagger)$. Suppose first that $\{s, t\} \cap V(T_{y,x}^\dagger) \neq \emptyset$, say $s \in V(T_{y,x}^\dagger)$ without loss of generality. If $s \in V(P_{y,z}) \setminus \{y, z\}$ then, by the choice of z and because S is a

8

$\mathcal{P}$-multicut, $P_{s,t}$ is intersected by $S \setminus V(T_{y,x}^{\dagger}) \subseteq S'$. Otherwise, $P_{s,t}$ passes through z and is therefore intersected by S'. Since it is clear that $P_{s,t}$ is intersected by S' if $\{s,t\} \cap V(T_{y,x}^{\dagger}) = \emptyset$, we conclude that S' is indeed a $\mathcal{P}_2$-multicut.

($\Leftarrow$) Suppose first that $\mathcal{I}_1$ is a YES-instance and let S_1 be a solution of $\mathcal{I}_1$. Since $\mathtt{wt}_1(y^\circ) = \mathbf{w} + 1$, $y^\circ \notin S$. This implies, in particular, that $(y^\circ, y^\circ) \notin \mathcal{P}_1$ and thus, no path of $\mathcal{P}$ is contained in $P_{y,x}$. Therefore, S_1 is a solution for $\mathcal{I}$. Suppose next that there exists $i \in \{0, \ldots, k\}$ such that $\mathcal{I}_{2,i}$ is a YES-instance and let $S_{2,i}$ be a minimal solution of $\mathcal{I}_{2,i}$. We first claim that $|S_{2,i} \cap V(P_{y,x})| = 1$. Indeed, observe that $S_{2,i} \cap V(P_{y,x}) \neq \emptyset$ since $(y,x) \in \mathcal{P}_2$. For the sake of contradiction, suppose that there exist $z, z' \in S_{2,i} \cap V(P_{y,x})$ such that $z' \neq z$, say z' is a descendant of z. Since, by construction, no path of $\mathcal{P}_2$ is contained in $T_{y,x}^{\dagger}$, each path of $\mathcal{P}_2$ that passes through z', also passes through z. Thus, $S_{2,i} \setminus \{z'\}$ is a $\mathcal{P}_2$-multicut, contradicting the minimality of $S_{2,i}$. Let $S_{2,i} \cap V(P_{y,x}) = \{z\}$. As argued above, if $z \neq x$, then, from the choice of x and y, every path of $\mathcal{P}$ contained in $T_{z,x}^{\dagger}$ passes through x. Similarly, every path of $\mathcal{P}$ contained in T_x passes through x. Let S^* be a $\mathcal{P}|_{T_{z,x}^{\dagger}}$-multicut of size at most i such that $\mathtt{wt}(S^*)$ is minimum. Then $\mathtt{wt}(S^*) = \mathcal{A}_{\mathsf{dpc}}(T_{z,x}^{\dagger}, \mathcal{P}_{T_{z,x}^{\dagger}}, x, \mathtt{wt}|_{T_{z,x}^{\dagger}}, i)$. Let $S = S_{2,i} \cup S^*$. We claim that S is a solution for $\mathcal{I}$. Indeed, first note that $|S| = |S_{2,i}| + |S^*| \leq k - i + i = k$. Furthermore, since $S_{2,i} \cap V(P_{y,x}) = \{z\}$, $\mathtt{wt}(S_{2,i}) = \mathtt{wt}_{2,i}(S_{2,i}) - \mathtt{wt}_{2,i}(z) + \mathtt{wt}(z)$ and $\mathtt{wt}_{2,i}(z) = \mathtt{wt}(z) + \mathtt{wt}(S^*)$. Thus, $\mathtt{wt}(S) = \mathtt{wt}(S_{2,i}) + \mathtt{wt}(S^*) \leq \mathtt{wt}_{2,i}(S_{2,i}) \leq \mathbf{w}$. We now show that S is a $\mathcal{P}$-multicut. Since $S_{2,i} \subseteq S$ and $S_{2,i}$ is a $\mathcal{P}_2$-multicut, any path of $\mathcal{P}$ fully contained in $V(T) \setminus V(T_{y,x}^{\dagger})$ is intersected by S. Consider now a path $P_{s,t}$ of $\mathcal{P}$ that intersects $V(T_{y,x})$ If $P_{s,t}$ is fully contained in $T_{z,x}^{\dagger}$, then it is intersected by S^*. Similarly, if $P_{s,t}$ passes through z, then it is intersected by S since $z \in S$. If $P_{s,t}$ passes through y without containing z, then $P_{s,t} \in \mathcal{P}_2$ and so, by the choice of z, $P_{s,t}$ is intersected by $S_{2,i} \setminus \{z\} \subseteq S$. Observe finally that $P_{s,t}$ is not fully contained in $V(P_{y,z}^{\dagger}) \setminus \{z\}$ for otherwise, $P_{s,t}$ is not intersected by X, a contradiction to the fact that X is a $\mathcal{P}$-multicut. Therefore, S is a solution for $\mathcal{I}$. $\qquad\square$

Proof of Theorem 1.1. Let $\mathcal{I} = (T, \mathcal{P}, \mathtt{wt}, \mathbf{w}, k)$ be an instance of wMC-TREE. Lemma 3.1 shows that the above algorithm correctly solves the problem. The described algorithm does a $(k+2)$-way branching, where the measure of the input instance is bounded by $2k$ and drops by at least 1 in every branch. Since the branching stops when the measure is at most 1, the total number of branching nodes of the algorithm is at most $(k+2)^{2k+1}$. Since $\mathcal{I}_1$ can be constructed in polynomial time and each instance $\mathcal{I}_{2,i}$ can be constructed by making $\mathcal{O}(n)$ calls to $\mathcal{A}_{\mathsf{dpc}}$, the final running time is $2^{\mathcal{O}(k^4)} \cdot n^{\mathcal{O}(1)}$. $\qquad\square$

4 uwMC-TREE Parameterized by the Number of Leaves

In this section, we prove Theorem 1.2. We first show that the problem on sub-divided stars can be solved without using the flow augmentation from [15] (Lemma 4.3). Towards this, we first design a simple polynomial-time algorithm for the problem on paths (Lemma 4.1) and use it to eliminate the terminal pair paths that do not pass through the high degree vertex of the sub-divided star. We then observe that the problem on sub-divided stars, when each terminal pair path pass through the high degree vertex, corresponds to the arcless instances of [15, Section 6.2.2], which can be solved faster [15, Lemma 6.12] (Proposition 4.2). We then use the algorithm of Lemma 4.3 as a subroutine to design a branching algorithm that proves Theorem 1.2.

Lemma 4.1. *Let T be a disjoint union of paths, $\mathcal{P} \subseteq V(T) \times V(T)$ and $\mathtt{wt} : V(T) \to \mathbb{N}$. There is an algorithm $\mathcal{A}_{\mathsf{path}}$ that outputs the weight of a $\mathcal{P}$-multicut $S \subseteq V(T)$, in T such that $\mathtt{wt}(S)$ is minimum, in polynomial time.*

Proof. If T is the union of at least two disjoint paths, then it is enough to solve the problem on each path independently and output the sum of the weights returned in each instance. Without loss of generality, let T be a path $(v_1, \ldots, v_n)$. For each $i \in [n]$, let $T_i = T[\{v_1, \ldots, v_i\}]$. We use dynamic programming to compute for each $i \in [n]$, $\mathsf{B}[i]$ which stores the minimum weight of a $\mathcal{P}|_{T_i}$-multicut in T_i. This is computed as follows. For any $i \in [n]$ such that $\mathcal{P}|_{T_i} \neq \emptyset$, let $i^* \leq i$ be the largest index such that there exists $(s, t) \in \mathcal{P}$ where $P_{s,t} \subseteq T[\{v_j \mid j \in \{i^*, \ldots, i\}\}]$. Then $\mathsf{B}[i]$ is computed as follows.

$$\mathsf{B}[i] = \begin{cases} 0 & \text{if } \mathcal{P}|_{T_i} = \emptyset \\ \mathtt{wt}(v_1) & \text{if } i = 1 \\ \min_{i^* \leq j \leq i}\{\mathtt{wt}(v_j) + \mathsf{B}[j-1]\} & \text{otherwise.} \end{cases}$$

The algorithm then returns $\mathsf{B}[n]$. Clearly, the algorithm runs in polynomial time. If $\mathcal{P}|_{T_i} = \emptyset$, then $S = \emptyset$ is a solution. Otherwise, if $i = 1$, then $(v_1, v_1) \in \mathcal{P}$ and so, $S = \{v_1\}$ is a minimum weight solution. If $i > 1$, then from the choice of i^*, for any minimal $\mathcal{P}|_{T_i}$-multicut S, $|S \cap \{v_{i^*}, \ldots, v_i\}| = 1$. If $S \cap \{v_{i^*}, \ldots, v_i\} = \{v_j\}$, then by induction, $\mathtt{wt}(S) = \mathtt{wt}(v_j) + \mathsf{B}[j-1]$. Again from the choice of i^*, for any $j \in \{i^*, \ldots, i\}$, v_j union any $\mathcal{P}|_{T_{j-1}}$-multicut is also a $\mathcal{P}|_{T_i}$-multicut. $\square$

An **r**-rooted subdivided star is a subdivided star with root **r**, where **r** is a highest degree vertex of T, that is, **r** is the unique degree 3 vertex, if it exists, or any arbitrary internal vertex if T is a path. Consider an instance $(T, \mathcal{P}, \mathbf{r}, \mathtt{wt})$ such that T is an **r**-rooted subdivided star and all the paths of $\mathcal{P}$ in T pass through **r**. The goal is to find a $\mathcal{P}$-multicut S such that $\mathtt{wt}(S)$ is minimum. We show that such instances corresponds to the *arcless* instances of wDPC as defined in [15, Section 6.2.2]. An instance $(T, \mathcal{P}, \mathtt{wt})$ is an arcless instance if (i) given two designated vertices s, t, the graph T consists of only internally vertex-disjoint paths from s to t, and (ii) if it is a YES-instance, then there exists a solution for this instance that intersects every (s, t)-path exactly once.

The following result follows from [15, Section 6.2.2, Lemma 6.12]. The root of a subdivided star, that is not a path, is the unique branching vertex.

Proposition 4.2 ([15]). *Given an instance $(T, \mathcal{P} \subseteq V(T) \times V(T), \mathbf{r} \in V(T), \mathit{wt} : V(T) \to \mathbb{N})$ such that T is a subdivided star with root $\mathbf{r}$ and $\ell \geq 3$ leaves. Suppose all the terminal pair paths in $\mathcal{P}$ pass through $\mathbf{r}$. Then one can find the weight of a $\mathcal{P}$-multicut $S \subseteq V(T)$ such that $\mathit{wt}(S)$ is minimum, in time $2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n^{\mathcal{O}(1)}$.*

Proof. We show how to reduce the given instance to an equivalent arcless instance of wDPC. Let $u_1, \ldots, u_\ell$ be the leaves of T and let P_i be the path from **r** to u_i. For each $i \in [\ell]$, guess whether the solution of $\mathcal{I}$ intersects $P_i \setminus \{\mathbf{r}\}$. This takes 2^ℓ branches. In the branch corresponding to the guess where $S \cap (V(P_i) \setminus \{\mathbf{r}\}) = \emptyset$, contract all the edges of P_i onto the vertex **r**. Observe that the resulting graph in each of the branches is still a subdivided star and has the property that the solution intersects every root-to-leaf path. Further note that no minimal solution contains more than one vertex from the root-to-leaf path, otherwise the deletion of a vertex that is furthest from the root would result in a smaller solution. Thus, there exists a minimal solution that intersects every root-to-leaf path exactly once. Let T' be constructed from T by (1) adding a new vertex $\bar{\mathbf{r}}$ and making it an out-neighbour of all the leaves of T, and (2) orienting every edge $xy \in E(T)$ from x to y if y is closer to $\bar{\mathbf{r}}$ than x (in the undirected setting). Then clearly T' is a graph consisting of internally vertex-disjoint $(\mathbf{r}, \bar{\mathbf{r}})$-paths such that the solution intersects each vertex-disjoint path exactly once. We define a weight function $\mathtt{wt}' : E(T') \to \mathbb{N}$ as follows. For every $i \in [\ell]$, let $P_i = (v_{i,1}, \ldots, v_{i,p_i})$ where $\mathbf{r} = v_{i,1}$ and $u_i = v_{i,p_i}$. Then $\mathtt{wt}'((v_{i,j}, v_{i,j+1})) = \mathtt{wt}(v_{i,j+1})$ for each $j \leq p_i - 1$, and $\mathtt{wt}'((v_{i,p_i}, \bar{\mathbf{r}})) = L$ where $L = \sum_{v \in V(T)} \mathtt{wt}(v) + 1$. Then observe that $V' = \{v_{1,i_1}, \ldots, v_{\ell,i_\ell}\}$ is a $\mathcal{P}$-multicut in T
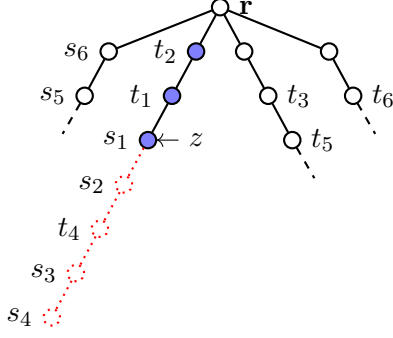
Figure 4: Updating the weights of the vertices in Lemma 4.3. $\mathcal{P} = \{(s_i, t_i) \mid i \in [6]\}$, z is closest to r such that the (s_1, t_1)-path is contained in the $(z, \mathbf{r})$-path.

The dotted parts are deleted and the weights of the filled vertices also include the weight of the minimum weight solution below them.

with $\mathtt{wt}(V') = \mathbf{w}$ if and only if $E' = \{(v_{1,i_1-1}, v_{1,i_1}), \ldots, (v_{\ell,i_\ell-1}, v_{\ell,i_\ell})\}$ is a $\mathcal{P}$-dpc in T' with $\mathtt{wt}'(E') = \mathbf{w}$. $\qquad\square$

Lemma 4.3. UWMC *can be solved in $2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n^{\mathcal{O}(1)}$-time on a subdivided star with ℓ leaves.*

Proof. Let the input instance be $\mathcal{I} = (T, \mathcal{P}, \mathtt{wt}, \mathbf{w})$. If $\ell = 2$, then T is a path. In this case, report YES if and only if $\mathcal{A}_{\mathsf{path}}(T, \mathcal{P}, \mathtt{wt}) \leq \mathbf{w}$. Otherwise, let $\mathbf{r} \in V(T)$ be the root of T, that is $\mathbf{r}$ is the unique vertex of degree at least 3 in T. In the first step, the algorithm guesses whether $\mathbf{r}$ is in the solution or not. If $\mathbf{r}$ belongs to the solution, then delete $\mathbf{r}$ from T and solve the resulting instance using $\mathcal{A}_{\mathsf{path}}$. Formally, the algorithm returns YES if and only if $\mathtt{wt}(\mathbf{r}) + \mathcal{A}_{\mathsf{path}}(T, \mathcal{P}, \mathtt{wt}) \leq \mathbf{w}$. Henceforth, we assume that the solution does not contain $\mathbf{r}$, or equivalently, we set $\mathtt{wt}(\mathbf{r}) = \mathbf{w} + 1$. The remaining algorithm has two phases. In the first phase, it eliminates all the paths in $\mathcal{P}$ that do not pass through $\mathbf{r}$. In the second phase, it uses the algorithm of Proposition 4.2 to solve the problem.

Suppose that there exists a path in $\mathcal{P}$ that does not pass through $\mathbf{r}$. Let $z \in V(T)$ be a vertex that is closest to $\mathbf{r}$ such that there exists a path $P_{s,t}$ in $\mathcal{P}$ where $P_{s,t} \subseteq P_{\mathbf{r},z}^\dagger$. We create a new instance $\mathcal{I}' = (T', \mathcal{P}', \mathtt{wt}', \mathbf{w})$ (in polynomial time) such that $\mathcal{I}'$ is equivalent to $\mathcal{I}$. Here, $T' = T \setminus T_z^\dagger$ and, $\mathcal{P}' = \mathcal{P} \setminus (V(T_{\mathbf{r},z}^\dagger) \times V(T_{\mathbf{r},z}^\dagger)) \cup \{(\mathbf{r}, z)\}$. Observe that the new terminal pair path $P_{\mathbf{r},z}$ in $\mathcal{P}'$ intersects $\mathbf{r}$ and thus, $\mathcal{P}'$ contains strictly fewer paths that do not pass through $\mathbf{r}$ (compared to $\mathcal{P}$). Since T is a subdivided star, for each $v \in V(T) \setminus \{\mathbf{r}\}$, $T_v^\dagger$ is a path. The new weight function $\mathtt{wt}'$ is defined as follows (see Figure 4).

$$\mathtt{wt}'(v) = \begin{cases} \mathtt{wt}(v) + \mathcal{A}_{\mathsf{path}}(T_v^\dagger, \mathcal{P}|_{T_v^\dagger}, \mathtt{wt}|_{V(T_v^\dagger)}) & \text{if } v \in V(P_{\mathbf{r},z}^\dagger) \\ \mathtt{wt}(v) & \text{otherwise.} \end{cases}$$

($\Rightarrow$) Let S be a $\mathcal{P}$-multicut of T such that $\mathtt{wt}(S) \leq \mathbf{w}$. Since $P_{\mathbf{r},z}^\dagger$ contains a path of $\mathcal{P}$, $S \cap V(P_{\mathbf{r},z}^\dagger) \neq \emptyset$. Let $y \in S \cap V(P_{\mathbf{r},z}^\dagger)$ be the vertex that is closest to $\mathbf{r}$. Construct $S' = S \setminus V(T_y^\dagger)$. We claim that S' is a solution for $\mathcal{I}'$. Observe that $S' \cap V(T_{\mathbf{r},z}^\dagger) = \{y\}$. Observe that $S \cap V(T_y^\dagger)$ is a $\mathcal{P}|_{T_y^\dagger}$-multicut in $T_y^\dagger$. Thus, $\mathtt{wt}(S \cap V(T_y^\dagger)) \geq \mathcal{A}_{\mathsf{path}}(T_y^\dagger, \mathcal{P}|_{T_y^\dagger}, \mathtt{wt}|_{V(T_y^\dagger)})$. From the construction of S' and the weight function $\mathtt{wt}'$, $\mathtt{wt}'(S') = \mathtt{wt}(S) - \mathtt{wt}(S \cap V(T_y^\dagger)) - \mathtt{wt}(y) + \mathtt{wt}'(y) \leq \mathtt{wt}(S) \leq \mathbf{w}$. We now show that S' is a $\mathcal{P}'$-multicut. Since $y \in S' \cap V(P_{\mathbf{r},z}^\dagger)$, $T - S'$ has no $(\mathbf{r}, z)$-path. Consider any path of $\mathcal{P}'$ that intersects a vertex of $T_y^\dagger$. Since the paths of $\mathcal{P}'$ are not contained in $T_{\mathbf{r},z}^\dagger$, such a path also pass through $\mathbf{r}$ and hence y. Since $y \in S'$, S' is a $\mathcal{P}'$-multicut.

($\Leftarrow$) Let S' be a minimal $\mathcal{P}'$-multicut in T such that $\mathtt{wt}'(S') \leq \mathbf{w}$. Then $T - S'$ has no $(\mathbf{r}, z)$-path. Since $\mathtt{wt}'(\mathbf{r}) = \mathbf{w} + 1$, $S' \cap V(P_{\mathbf{r},z}^\dagger) \neq \emptyset$. Since S' is a minimal solution, $|S' \cap V(P_{\mathbf{r},z}^\dagger)| = 1$ for otherwise, deleting the vertex of S on the $(\mathbf{r}, z)$-path that is furthest from $\mathbf{r}$ would result in a smaller solution. Let $S' \cap V(P_{\mathbf{r},z}^\dagger) = \{y\}$. Let S^* be a minimum weight $\mathcal{P}|_{T_y^\dagger}$-multicut. Then $\mathtt{wt}(S^*) = \mathcal{A}_{\mathsf{path}}(T_y^\dagger, \mathcal{P}|_{T_y^\dagger}, \mathtt{wt}|_{V(T_y^\dagger)})$. Construct $S = S' \cup S^*$. We will now show that S is a solution of $\mathcal{I}$. From the construction of S and $\mathtt{wt}'$, $\mathtt{wt}(S) = \mathtt{wt}(S') + \mathtt{wt}(S^*) = \mathtt{wt}'(S') - \mathtt{wt}'(y) +$

11

$\mathtt{wt}(y) + \mathtt{wt}(S^*) \le \mathtt{wt}'(S') \le \mathbf{w}$. Since $S' \subseteq S$, S is a $\mathcal{P}'$-multicut. Consider a path of $\mathcal{P}$ that is contained in $T^{\dagger}_{\mathbf{r},z}$. If such a path passes through y or is contained in T_y, then it is intersected by $S^* \cup \{y\}$ (and hence S). Otherwise such a path is contained in $P^{\dagger}_{\mathbf{r},y} \setminus \{y\}$. But this contradicts the choice of z. Therefore, S is indeed a $\mathcal{P}$-multicut.

We conclude that whenever there exists a path in $\mathcal{P}$ that does not pass through $\mathbf{r}$, we can apply the above procedure in polynomial time. Since every application of the above procedure decreases the number of paths of $\mathcal{P}$ that do not pass through $\mathbf{r}$ by at least one, the above procedure can be exhaustively applied in polynomial time. This ends the first phase of the algorithm. At the end of the first phase, all the paths of $\mathcal{P}$ pass through $\mathbf{r}$. Therefore, in this case, we solve the instance $(T, \mathcal{P}, \mathbf{r}, \mathtt{wt})$ using the algorithm of Proposition 4.2. Since the first phase of the algorithm takes polynomial time and the second phase takes $2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n^{\mathcal{O}(1)}$-time, the algorithm runs in time $2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n^{\mathcal{O}(1)}$. $\qquad\square$

Observe that we can use Lemma 4.3 to find the minimum weight $\mathcal{P}$-multicut in a subdivided star by doing a simple binary search starting with $\mathbf{w} = 0, 1, 2, 4, 8, \dots$ and so on. This would incur an extra $\mathcal{O}(\log \mathbf{w})$ factor in the running time. Thus, even if $\mathbf{w}$ is given as a unary input, the resulting algorithm is still polynomial in the input size. Therefore, the following corollary follows from Lemmas 4.1 and 4.3.

Corollary 4.4. *Let T be a subdivided star with ℓ leaves. Let $\mathcal{P} \subseteq V(T) \times V(T)$ and $\mathtt{wt} : V(T) \to \mathbb{N}$. There is an algorithm $\mathcal{A}_{\mathsf{star}}$ that finds the weight of a $\mathcal{P}$-multicut $S \subseteq V(T)$ such that $\mathtt{wt}(S)$ is minimum, in $2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n^{\mathcal{O}(1)}$-time.*

We are now equipped to design the branching algorithm for Theorem 1.2. Let $\mathcal{I} = (T, \mathcal{P}, \mathtt{wt}, \mathbf{w})$ be an instance of UWMC-TREE. Root T at an arbitrary vertex $\mathbf{r}$. With each instance $\mathcal{I}$, we associate the measure $\mu(\mathcal{I}) = |V_{\ge 3}(T)| + |V_{=1}(T)|$. Since $|V_{=1}(T)| \le \ell$ and $|V_{\ge 3}(T)| \le |V_{=1}(T)| - 1$, $\mu(\mathcal{I}) \le 2\ell$. We now design a branching algorithm such that the measure μ drops in each branch. The following cases appear as base cases: (1) If $|V_{\ge 3}(T)| \le 1$, then return YES if and only if $\mathcal{A}_{\mathsf{star}}(T, \mathcal{P}, \mathtt{wt}) \le \mathbf{w}$, and (2) If $\mathbf{w} < 0$ or, $\mathbf{w} \le 0$ and $\mathcal{P} \ne \emptyset$, then return NO.
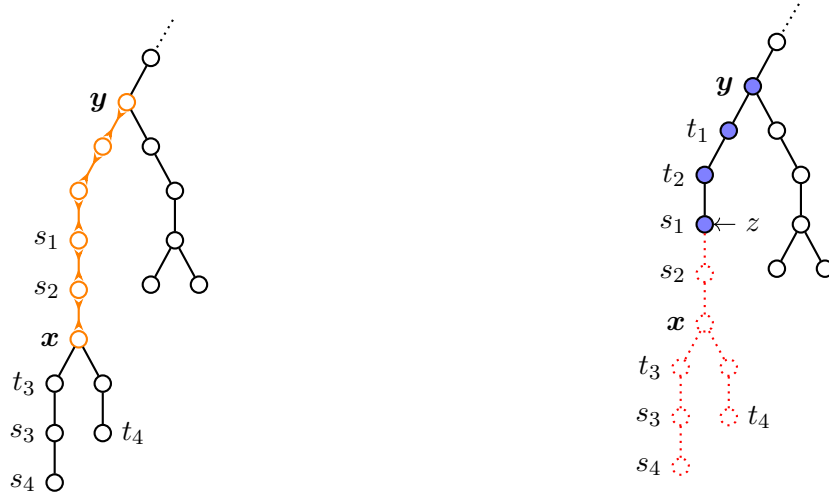
If $|V_{\ge 3}(T)| \ge 2$, let $x, y \in V_{\ge 3}(T)$ such that x is a furthest in T and, y is its unique closest ancestor. We branch into the following two cases.

Case 1. *There exists a solution of $\mathcal{I}$ that does not intersect $V(P_{y,x})$.* In this branch, we return the instance $\mathcal{I}_1 = (T_1, \mathcal{P}_1, \mathtt{wt}_1, \mathbf{w})$ where $T_1 = T/E(P_{y,x})$ and $\mathcal{P}_1 = \mathcal{P}/E(P_{y,x})$. Let the vertex onto which the edges of $P_{y,x}$ are contracted be y°. The new weight function $\mathtt{wt}_1$ is defined as follows: $\mathtt{wt}_1(v) = \mathtt{wt}(v)$ for each $v \in V(T_1) \setminus \{y^{\circ}\}$, and $\mathtt{wt}_1(y^{\circ}) = \mathbf{w} + 1$. Observe that $\mathcal{I}_1$ can be constructed in polynomial time. Furthermore, since $x, y \in V_{\ge 3}(T)$ and the edges of $P_{y,x}$ are contracted in $\mathcal{I}_1$, $|V_{\ge 3}(T_1)| = |V_{\ge 3}(T)| - 1$ and thus, $\mu(\mathcal{I}_1) = \mu(\mathcal{I}) - 1$.

Case 2. *There exists a solution of $\mathcal{I}$ that intersects $V(P_{y,x})$.* In this case, let $z \in V(P_{y,x})$ be the closest vertex to y such that $P_{y,z}$ contains a path of $\mathcal{P}$. If no such vertex exists then set $z = x$. Return the instance $\mathcal{I}_2 = (T_2, \mathcal{P}_2, \mathtt{wt}_2, \mathbf{w})$ where $T_2 = T \setminus T^{\dagger}_z$ and $\mathcal{P}_2 = (\mathcal{P} \setminus (V(T_{y,x}) \times V(T_{y,x}))) \cup \{(y, z)\}$. Observe that, by construction, any solution of $\mathcal{I}$ intersects $V(P_{y,z})$. The new weight function $\mathtt{wt}_2$ is defined as follows (see Figure 5).

$$\mathtt{wt}_2(v) = \begin{cases} \mathtt{wt}(v) + \mathcal{A}_{\mathsf{star}}(T^{\dagger}_{v,x}, \mathcal{P}|_{T^{\dagger}_{v,x}}, \mathtt{wt}|_{V(T^{\dagger}_{v,x})}) & \text{if } v \in V(P_{y,z}) \\ \mathtt{wt}(v) & \text{otherwise.} \end{cases}$$

Observe that, for each $v \in V(P_{y,x}) \setminus \{x\}$, $T^{\dagger}_{v,x}$ has exactly one branching vertex, namely x, since x is a furthest branching vertex in T from $\mathbf{r}$, y is the branching vertex that is the closest ancestor of x and $v \in V(P_{y,x})$. Also, $T^{\dagger}_{x,x} = T^{\dagger}_x$ is a disjoint union of paths. Since $x \in V_{\ge 3}(T)$, from the construction of T_2, $|V_{=1}(T_3)| < |V_{=1}(T)|$ and so, $\mu(\mathcal{I}_2) < \mu(\mathcal{I})$.

(a) Case 1. The marked orange edges are contracted onto the undeletable vertex y°.

(b) Case 3. z is closest to y such that the (y, z)-path contains (s_1, t_1). The weight of the filled vertices includes the weight of the minimum weight solution below them. $T_z^\dagger$ is deleted (dotted part).

Figure 5: The branches of the algorithm of Theorem 1.2 with terminal pairs (s_i, t_i). x is a furthest branching vertex and y its unique closest branching ancestor.

Lemma 4.5. *$\mathcal{I}$ is a YES-instance if and only if $\mathcal{I}_1$ or $\mathcal{I}_2$ is a YES-instance.*

Proof. ($\Rightarrow$) Let $\mathcal{I}$ be a YES-instance and let S be a solution of $\mathcal{I}$. Suppose first that $S \cap V(P_{y,x}) = \emptyset$ and recall that y° is the vertex onto which the path $P_{y,x}$ is contracted in $\mathcal{I}_1$. Consider a path $P_{s,t}$ in $\mathcal{P}_1$. Then $(s, t) \neq (y^\circ, y^\circ)$ for otherwise, S would not intersect the path in $\mathcal{P}$ corresponding to (s, t). If $y^\circ \notin \{s, t\}$ then $(s, t) \in \mathcal{P}$ and so, S intersects the path $P_{s,t}$. Otherwise, assume, without loss of generality, that $s = y^\circ$ and let $(z, t) \in \mathcal{P}$ where $z \in V(P_{y,x})$, be the terminal pair in $\mathcal{P}$ corresponding to (s, t). Then since $P_{z,t}$ is intersected by $S \setminus V(P_{y,x})$, we conclude that $P_{s,t}$ is also intersected by $S \setminus \{y^\circ\}$.

Now suppose that $S \cap V(P_{y,x}) \neq \emptyset$. From the choice of z, $S \cap V(P_{y,z}) \neq \emptyset$. Let v be the closest vertex of $P_{y,z}$ to y that belongs to S. Construct $S' = S \setminus V(T_{v,x}^\dagger)$. We claim that S' is a $\mathcal{P}_2$-multicut in T_2 and $\mathtt{wt}_2(S') \leq \mathbf{w}$. Since $v \in V(P_{y,z}) \cap S'$, $T_2 - S'$ does not contain the (y, z)-path. Consider now a path $P_{s,t}$ in $\mathcal{P}_2 \setminus \{(y, z)\}$. Then by definition of $\mathcal{P}_2$, $|\{s, t\} \cap V(T_{y,x})| \leq 1$. If $\{s, t\} \cap V(T_{y,x}) = \emptyset$ then $P_{s,t}$ is intersected by $S \setminus V(T_{y,x}) \subseteq S'$ since S is a $\mathcal{P}$-multicut. Thus, suppose that $\{s, t\} \cap V(T_{y,x}) \neq \emptyset$, say $s \in V(T_{y,x})$ without loss of generality (note that then, $t \in V(T) \setminus V(T_{y,x})$). If $P_{s,t}$ contains v, then $P_{s,t}$ is intersected by S' since $v \in S'$. Otherwise, $s \in V(P_{y,v}) \setminus \{v\}$ in which case, by the choice of v and because S is a $\mathcal{P}$-multicut, $P_{s,t}$ is intersected by $S \setminus V(T_{y,x}) \subseteq S'$. Thus, we conclude that S' is a $\mathcal{P}_2$-multicut in T_2. From the construction of S', observe that $S' \cap V(P_{y,z}) = \{v\}$. Thus, $\mathtt{wt}_2(S') = \mathtt{wt}(S) - \mathtt{wt}(S \cap V(T_{v,x}^\dagger)) - \mathtt{wt}(v) + \mathtt{wt}_2(v)$. Since $S \cap V(T_{v,x}^\dagger)$ is a $\mathcal{P}|_{T_{v,x}^\dagger}$-multicut, $\mathtt{wt}(S \cap V(T_{v,x}^\dagger)) \geq \mathcal{A}_{\mathsf{star}}(T_{v,x}^\dagger, \mathcal{P}_{T_{v,x}^\dagger}, \mathtt{wt}|_{V(T_{v,x}^\dagger)})$. Therefore, $\mathtt{wt}_2(S') \leq \mathtt{wt}(S) \leq \mathbf{w}$.

($\Leftarrow$) If $\mathcal{I}_1$ is a YES-instance, then since $\mathtt{wt}_1(y^\circ) = \mathbf{w} + 1$, no solution of $\mathcal{I}_1$ contains y°. Thus, every solution of $\mathcal{I}_1$ is also a solution for $\mathcal{I}$. If $\mathcal{I}_2$ is a YES-instance and let S' be a minimal solution of $\mathcal{I}_2$. Since S' is a $\mathcal{P}_2$-multicut and $(y, z) \in \mathcal{P}_2$, $S' \cap V(P_{y,z}) \neq \emptyset$. In fact, since S' is a minimal, $|S' \cap V(P_{y,z})| = 1$ for otherwise, deleting the vertex of S' on the (y, z)-path that is furthest from y would result in a smaller solution (recall that $\mathcal{P}_2$ contains no terminal pair in $V(T_{y,x}) \times V(T_{y,x})$). Let $v \in S \cap V(P_{y,z})$. If $v = x$, then S' is also a solution for $\mathcal{I}$. Otherwise, let $v \neq x$. Let S^* be the solution given by $\mathcal{A}_{\mathsf{star}}(T_{v,x}^\dagger, \mathcal{P}|_{T_{v,x}^\dagger}, \mathtt{wt}|_{V(T_{v,x}^\dagger)})$ and let $S = S' \cup S^*$. We claim that S is a $\mathcal{P}$-multicut in T such that $\mathtt{wt}(S) \leq \mathbf{w}$. Since $S' \subseteq S$, S is a $\mathcal{P}_2$-multicut.
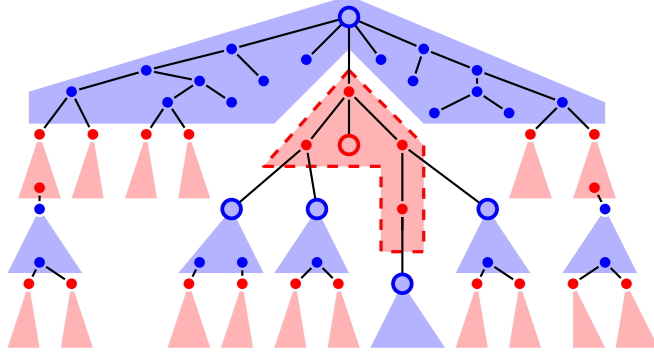
13

Figure 6: A $(d, 6)$-light instance for some d. The d-light vertices vertices are shown in blue, and the d-heavy vertices in red. The closed neighborhood of the central component (marked with dashed boundary) containing d-heavy vertices has six leaves (marked with big circles), five of which are d-light vertices.

Consider a path $P_{s,t}$ of $\mathcal{P}$ that is contained in $T_{y,x}$. Then either $P_{s,t}$ is contained in $T_{v,x}^\dagger$ or $P_{s,t}$ contains v: indeed, if neither hold then $P_{s,t}$ is contained in $P_{y,v} \setminus \{v\}$, a contradiction to the choice of z. Now if $P_{s,t}$ is contained in $T_{v,x}^\dagger$, then it is intersected by S^*; and if $P_{s,t}$ contains v, then it is intersected by S since $v \in S$. Thus, S is a $\mathcal{P}$-multicut. From the construction of S, $\mathtt{wt}(S) = \mathtt{wt}(S') + \mathtt{wt}(S^*)$. Also, $\mathtt{wt}(S') = \mathtt{wt}_2(S') - \mathtt{wt}_2(v) + \mathtt{wt}(v)$ because $S' \cap V(P_{y,z}) = \{v\}$. Since $\mathtt{wt}_2(v) = \mathtt{wt}(v) + \mathtt{wt}(S^*)$, we conclude that $\mathtt{wt}(S) = \mathtt{wt}_2(S') \leq \mathbf{w}$. $\qquad\square$

We now prove Theorem 1.2 formally.

Proof of Theorem 1.2. Let $\mathcal{I} = (T, \mathcal{P}, \mathtt{wt}, \mathbf{w})$ be an instance of UWMC-TREE. Lemma 4.5 shows that the algorithm described above correctly decides if T has a $\mathcal{P}$-multicut S such that $\mathtt{wt}(S) \leq \mathbf{w}$. Since the algorithm is a 2-way branching algorithm, the measure of the algorithm, which drops by one in each branching step, is bounded by 2ℓ and the branching stops when the measure is at most 1, the total number of nodes of the branching tree is at most $2^{2\ell+1}$. Now note that the worst running time at each branching node is during the construction of $\mathcal{I}_2$. However, since the construction of $\mathcal{I}_2$ requires making $\mathcal{O}(n)$ calls to the algorithm of Corollary 4.4 ($\mathcal{A}_{\mathsf{star}}$), $\mathcal{I}_2$ can be constructed in $2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n^{\mathcal{O}(1)}$ time. Thus, the algorithm runs in time $2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n^{\mathcal{O}(1)}$. $\qquad\square$

5 Weighted Multicut on (d, ℓ)-Light Instances

In this section we prove Theorem 1.3 by giving an FPT-algorithm which solves UWMC on (d, ℓ)-light instances in time $3^d \cdot 2^{d\ell} \cdot 2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n^{\mathcal{O}(1)}$. For this, we first formally define the notion of (d, ℓ)-light.

Definition 5.1. *Let T be a tree and $\mathcal{P} \subseteq V(T) \times V(T)$ be a set of terminal pairs. A vertex $v \in V(T)$ is called a d-light vertex of $(T, \mathcal{P})$ if at most d terminal pair paths of $\mathcal{P}$ pass through v in T.*

The set of d-light vertices of $(T, \mathcal{P})$ is denoted by $\mathsf{light}(T, \mathcal{P}, d)$.

We say that $(T, \mathcal{P})$ is (d, ℓ)-light if T is a tree and for each connected component C of $T - \mathsf{light}(G, \mathcal{P}, d)$, $N[C]$ has at most ℓ leaves.

For ease of notation, we say that a vertex $v \in V(T) \setminus \mathsf{light}(G, \mathcal{P}, d)$ is d-heavy. See Figure 6 for an illustration of this definition.

In the definition of (d, q)-light instances, it is crucial to consider the number of leaves in the tree induced by the *closed neighborhood* of each component C of $T - Y$ (i.e. $T[N[C]]$ must have at most q leaves). Assume, for a moment, that we just require that $T[C]$ has at most q leaves, then we do not expect the result as in Theorem 1.3.

14

This is because with this new (and wrong) definition of (d, q)-light instances, UWMC-TREE is NP-hard for $d = 3$ and $q = 2$. Let (G, k) be an instance of VERTEX COVER with $V(G) = \{v_1, \ldots, v_n\}$. Let G' be a graph on $2n$ vertices $x_1, \ldots, x_n, y_1, \ldots, y_n$ where x_i is adjacent to x_{i+1} for all $i \in [n-1]$ and each y_i adjacent to x_i for all $i \in [n]$. Define the set of terminal pairs $\mathcal{P}$ as $(y_i, y_j) \in \mathcal{P}$ if and only if $(v_i, v_j) \in E(G)$. The weight function $\mathtt{wt} : V(G') \to \{1, n+1\}$ is defined as $\mathtt{wt}(x_i) = n + 1$ for each $i \in [n]$ and $\mathtt{wt}(y_i) = 1$ for each $i \in [n]$. It is easy to observe that $\{v_{i_1}, \ldots, v_{i_k}\}$ is a vertex cover of G if and only if $\{y_{i_1}, \ldots, y_{i_k}\}$ is a $\mathcal{P}$-multicut in G'. We know that VERTEX COVER is NP-hard on graphs with maximum degree 3 [10]. Hence, we can assume that for each y_i at most 3 terminal pair paths of $\mathcal{P}$ pass through it. Thus $Y = \{y_1, \ldots, y_n\}$ is a set of 3-light vertices and the removal of any superset of it leaves a collection of paths each of which has at most 2 leaves. Thus, with the *wrong* definition of (d, q)-light instances, the resulting instance is a $(3, 2)$-light instance.

Notation. Let $\mathcal{I} = (T, \mathcal{P}, \mathtt{wt}, \mathbf{w})$ be a UWMC instance. Then $\mathcal{I}$ is called a (d, ℓ)-light instance if $(T, \mathcal{P})$ is (d, ℓ)-light.

Assume that T is rooted. For every vertex $v \in V(T)$, we denote by $I[v] \subseteq \mathcal{P}|_{T_v}$ the set of terminal pairs $(s, t) \in \mathcal{P}|_{T_v}$ such that v is contained in the (s, t)-path in T, and by $O[v] \subseteq \mathcal{P}$ the set of terminal pairs $(s, t) \in \mathcal{P}$ such that $\{s, t\} \cap V(T_v) \neq \emptyset$ and $\{s, t\} \cap V(T) \setminus V(T_v) \neq \emptyset$. In other words, $I[v]$ denotes the set of terminal pairs going through v and which are fully contained in the subtree rooted at v. In constrast, the set $O[v]$ contains those terminal pairs going through v and leaving the subtree rooted at v. Note that if v is a d-light vertex then $|I[v]| + |O[v]| \leq d$ by definition.

Intuition. The intuition of the algorithm is as follows. For each d-light vertex $v \in V(T)$ and for all sets $O \subseteq O[v]$, we compute the minimum weight of a partial solution $S_{O,v} \subseteq V(T_v)$ such that $S_{O,v}$ is a $\mathcal{P}|_{T_v}$-multicut and for every $(s, t) \in O$, $S_{O,v}$ intersects the (s, t)-path in T. We store this minimum weight of a solution as $\mathsf{Tab}[v, O]$. We use a dynamic program to compute the table entries for the d-light vertices in a bottom-up transversal of T (we assume that T is rooted). The crucial part of the algorithm is that we do *not compute these partial solutions* for *every d-heavy vertices*. Instead, one can think of partitioning the tree into (connected) components corresponding to the status of being d-light or d-heavy. For the components with the d-light vertices, we compute the best solution by an exhaustive search. This works because there are at most d terminal pair paths going through a d-light vertex. For the components consisting of d-heavy vertices, we make use of the previous result in Theorem 1.2 to compute a minimum solution. Here, we exploit the fact that such a component has at most ℓ leaves. We first design the main algorithm that computes the table entries $\mathsf{Tab}[v, \cdot]$ for every $v \in \mathsf{light}(T, \mathcal{P}, d)$. This algorithm uses as a subroutine the second algorithm $\mathcal{A}_v^{\mathsf{heavy}}$ to compute partial solutions for the d-heavy children of v.

First observe that if $(T, \mathcal{P})$ contains no d-light vertex then, by definition, T has at most ℓ leaves and thus, we may use the algorithm of Theorem 1.2 to solve UWMC on instance $\mathcal{I}$ in time $2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n^{\mathcal{O}(1)}$. Assume henceforth that $\mathsf{light}(T, \mathcal{P}, d) \neq \emptyset$ and let us root T at some d-light vertex $\mathbf{r}$. We define the table Tab formally as follows: for every $v \in \mathsf{light}(T, \mathcal{P}, d)$ and for every set $O \subseteq O[v]$,

$$\mathsf{Tab}[v, O] \overset{\text{def}}{=} \min_{S \subseteq V(T_v)} \mathtt{wt}(S) \text{ s.t.}$$

$$S \text{ is a } \mathcal{P}|_{T_v}\text{-multicut} \wedge \forall (s, t) \in O : S \text{ intersects the } (s, t)\text{-path in } T$$

Initially, every entry of the table Tab is set to $+\infty$. To update each entry of $\mathsf{Tab}[v, \cdot]$, we assume that $\mathcal{A}_v^{\mathsf{heavy}}$ is given. The output is YES if $\mathsf{Tab}[\mathbf{r}, \emptyset] \leq \mathbf{w}$ and NO otherwise. Note that this entry is defined as we assume that $\mathbf{r}$ is d-light. For every d-light vertex, we proceed as follows.

Updating d-Light Leaves. Let $v \in \mathsf{light}(T, \mathcal{P}, d)$ be a leaf of T. Then for every $O \subseteq O[v]$, we set

$$\mathsf{Tab}[v, O] = \begin{cases} \mathtt{wt}(v) & \text{if } O \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

Updating Internal d-Light Vertices. Let $v \in \mathsf{light}(T, \mathcal{P}, d)$ be an internal vertex of T and let $u_1, \ldots, u_p \in V(T)$ be the children of v. Let $O \subseteq O[v]$ be fixed. As mentioned above, we assume that there is a subroutine $\mathcal{A}_v^{\mathsf{heavy}}$ which takes as an input a child $u \notin \mathsf{light}(T, \mathcal{P}, d)$ of v and a set $Q \subseteq O[u]$ of terminal pairs, and outputs the minimum weight of a set $S \subseteq V(T_u)$ such that S is a $\mathcal{P}|_{T_u}$-multicut and for every $(s, t) \in Q$, S intersects the (s, t)-path in T (we show below how to obtain $\mathcal{A}_v^{\mathsf{heavy}}$). For ease of notation, we define a function $\mathcal{A}_v^*$: for every child u of v and every set $Q \subseteq O[u]$,

$$\mathcal{A}_v^*(u, Q) = \begin{cases} \mathsf{Tab}[u, Q] & \text{if } u \in \mathsf{light}(T, \mathcal{P}, d), \\ \mathcal{A}_v^{\mathsf{heavy}}(u, Q) & \text{otherwise.} \end{cases}$$

Intuition. We first describe the intuition of the algorithm. Note that it is always possible to delete v. In this case, the solution is the disjoint union of optimal solutions for the children, where we do not have to cut any of the outgoing terminal pairs. Moreover, we have to delete v if (v, v) is a terminal pair, or there are terminal pairs in O which start at v and leave T_v (these pairs are later denoted by O_0).

In the case where v is not deleted, we proceed as follows. We denote by $I_{i,j}$ the set of terminal pairs in $I[v]$ which use u_i and u_j where $i < j$. Likewise, the set $I_i \subseteq I[v]$ denotes the terminal pairs which use only u_i and end at v, i.e. they do not go into any other subtree. We guess which of the paths in $I_{i,j}$ are cut by the solution for the subtree rooted at u_i. We denote this set by $Q_{i,j}$. Note that the pairs in $I_{i,j} \setminus Q_{i,j}$ must then be cut by the solution for the subtree rooted at u_j. Besides these pairs, we also have to cut the terminal pairs which leave T_v and start/end in a subtree of some child u_i. We denote this set by O_i. Thus, for each child u_i we have to cut the terminal pairs in $O_i \cup I_i \cup \bigcup_{j>i} Q_{i,j} \cup \bigcup_{j<i} I_{j,i} \setminus Q_{j,i}$.

We now give the formal algorithm. For every $i, j \in [p]$ where $i < j$, denote by $I_{i,j} = I[v] \cap O[u_i] \cap O[u_j]$ and for every $i \in [p]$, denote by $I_i = I[v] \cap O[u_i] \setminus \bigcup_{j>i} I_{i,j}$. Further denote by $I_0 = \{(v, v)\} \cap I[v]$. Note that for every $(s, t) \in I[v]$, one of three cases may arise:

- $(s, t) \notin \bigcup_{i \in [p]} O[u_i]$, that is, $(s, t) = (v, v)$, or

- there exists a unique index $i \in [p]$ such that $(s, t) \in O[u_i]$, or

- there exist exactly two indices $i \in [p]$ such that $(s, t) \in O[u_i]$.

Indeed, the (s, t)-path would otherwise leave T_v thereby contradicting the fact that $(s, t) \in I[v]$. Therefore, $\{I_0, I_1, \ldots, I_p, I_{1,2}, \ldots, I_{p-1,p}\}$ is a partition $I[v]$.

Denote by $O_0 \subseteq O$ the set of terminal pairs $(s, t) \in O$ such that $(s, t) \notin \bigcup_{i \in [p]} O[u_i]$, and for every $i \in [p]$, denote by $O_i \subseteq O$ the set of terminal pairs $(s, t) \in O$ such that $(s, t) \in O[u_i]$. Note that for every $(s, t) \in O$, one of two cases may arise:

- $(s, t) \in O[v] \setminus \bigcup_{i \in [p]} O[u_i]$, or

- there exists a unique index $i \in [p]$ such that $(s, t) \in O[u_i]$.

Indeed, the (s, t)-path would otherwise not leave T_v thereby contradicting the fact that $(s, t) \in O[v]$. Therefore, $\{O_0, \ldots, O_p\}$ is a partition O.

If $I_0 \cup O_0 \neq \emptyset$, then the only way to separate the terminal pairs in $I_0 \cup O_0$ is by removing v. Thus, in this case, we update $\mathsf{Tab}[v, O]$ as follows.

$$\mathsf{Tab}[v, O] = \mathtt{wt}(v) + \sum_{i \in [p]} \mathcal{A}_v^*(u_i, \emptyset).$$

Otherwise, let a *distribution* be a $p(p-1)/2$-tuple $\pi = (Q_{1,2}, Q_{1,3}, \ldots, Q_{p-1,p})$ where for every $i \in [p-1]$ and $j > i$, $Q_{i,j}$ is a subset of $I_{i,j}$. Then we update $\mathsf{Tab}[v, O]$ according to the following procedure.

1. Set $\mathsf{Tab}[v, O] = \mathtt{wt}(v) + \sum_{i \in [p]} \mathcal{A}_v^*(u_i, \emptyset)$.

2. For every distribution $\pi = (Q_{1,2}, Q_{1,3}, \ldots, Q_{p-1,p})$ do:

    - For every $i \in [p]$, define $\mathcal{P}_i^\pi = O_i \cup I_i \cup \bigcup_{i<j} Q_{i,j} \cup \bigcup_{i>j} I_{j,i} \setminus Q_{j,i}$.
    - $\mathsf{Tab}[v, O] = \min\{\mathsf{Tab}[v, O], \sum_{i \in [p]} \mathcal{A}_v^*(u_i, \mathcal{P}_i^\pi)\}$

Lemma 5.2. *For every internal d-light vertex v, if $\mathcal{A}_v^*$ is correct then the table entries $\mathsf{Tab}[v, \cdot]$ are updated correctly. Furthermore, $\mathsf{Tab}[v, \cdot]$ can be updated in $3^d \cdot \mathcal{O}(\mathsf{T}(\mathcal{A}_v^{heavy}))$-time where $\mathsf{T}(\mathcal{A}_v^{heavy})$ is the running time of the subroutine $\mathcal{A}_v^{heavy}$.*

Proof. Let $v \in \mathsf{light}(T, \mathcal{P}, d)$ be an internal vertex of T and let $u_1, \ldots, u_p \in V(T)$ be the children of v. Assume that $\mathcal{A}_v^*$ is correct (in particular, for every child $u \in \mathsf{light}(T, \mathcal{P}, d)$ of v, the table $\mathsf{Tab}[u, \cdot]$ is correctly filled).

Let us first show that for any set $O \subseteq O[v]$, there exists a set $S \subseteq V(T_v)$ of weight $\mathsf{Tab}[v, O]$ such that S is a $\mathcal{P}|_{T_v}$-multicut and for every $(s, t) \in O$, S intersects the (s, t)-path in T.

Consider a set $O \subseteq O[v]$. Observe that the update step sets $\mathsf{Tab}[v, O]$ to a finite value. Suppose first that $I_0 \cup O_0 \neq \emptyset$. Since $\mathcal{A}_v^*$ is correct, it follows from the update step that there exists for every $i \in [p]$, a $\mathcal{P}|_{T_{u_i}}$-multicut S_i such that $\mathsf{Tab}[v, O] = \mathtt{wt}(v) + \sum_{i \in [p]} \mathtt{wt}(S_i)$. Then the set $\{v\} \cup \bigcup_{i \in [p]} S_i$ is the desired S. Second, suppose that $I_0 \cup O_0 = \emptyset$. Since $\mathcal{A}_v^*$ is correct, it follows from the update step that either

(i) there exists for every $i \in [p]$, a $\mathcal{P}|_{T_{u_i}}$-multicut S_i such that $\mathsf{Tab}[v, O] = \mathtt{wt}(v) + \sum_{i \in [p]} \mathtt{wt}(S_i)$, or

(ii) there is a distribution $\pi = (Q_{1,2}, Q_{1,3}, \ldots, Q_{p-1,p})$ such that for every $i \in [p]$, there exists a set S_i where

    - S_i is a $\mathcal{P}|_{T_{u_i}}$-multicut and
    - for every $(s, t) \in \mathcal{P}_i^\pi$, S_i intersects the (s, t)-path in T,

    and $\mathsf{Tab}[v, O] = \sum_{i \in [p]} \mathtt{wt}(S_i)$.

If (i) holds then we conclude, as previously, that $\{v\} \cup \bigcup_{i \in [p]} S_i$ is the desired S. Thus, assume that (ii) holds and let us show that $\bigcup_{i \in [p]} S_i$ is the desired S. Note that since for every $i \in [p]$, S_i is a $\mathcal{P}|_{T_{u_i}}$-multicut, it suffices to show that for every $(s, t) \in I[v] \cup O$, $\bigcup_{i \in [p]} S_i$ intersects the (s, t)-path in T. Consider, therefore, a terminal pair $(s, t) \in I[v] \cup O$. If $(s, t) \in O_i \cup I_i$ for some $i \in [p]$, then S_i intersects the (s, t)-path in T by definition. Thus, assume that $(s, t) \in I_{i,j}$ for some $i, j \in [p]$ where $i < j$. Then either $(s, t) \in Q_{i,j}$ in which case S_i intersects the (s, t)-path in T by definition; or $(s, t) \in I_{i,j} \setminus Q_{i,j}$ in which case S_j intersects the (s, t)-path in T by definition.

Second, let us show that for any set $O \subseteq O[v]$, if S is a set of minimum weight such that S is a $\mathcal{P}|_{T_v}$-multicut and for every $(s, t) \in O$, S intersects the (s, t)-path in T, then $\mathtt{wt}(S) \geq \mathsf{Tab}[v, O]$.

Let S be a set of minimum weight such that S is a $\mathcal{P}|_{T_v}$-multicut and for every $(s, t) \in O$, S intersects the (s, t)-path in T. For every $i \in [p]$, denote by $S_i = S \cap V(T_{u_i})$. Suppose first that

$v \in S$. Then for every $i \in [p]$, S_i is a $\mathcal{P}|_{T_{u_i}}$-multicut which implies that $\mathtt{wt}(S_i) \geq \mathcal{A}_v^*(u_i, \emptyset)$ as $\mathcal{A}_v^*$ is correct. But $\mathsf{Tab}[v, O] \leq \mathtt{wt}(v) + \sum_{i \in [p]} \mathcal{A}_v^*(u_i, \emptyset)$ by the update step and thus, the claim holds true. Second, suppose that $v \notin S$ (note that $I_0 \cup O_0 = \emptyset$ in this case). Then for every $i, j \in [p]$ where $i < j$, and every terminal pair $(s, t) \in I_{i,j}$, at least one of S_i and S_j must intersect the (s, t)-path in T: let us denote by $Q_{i,j} \subseteq I_{i,j}$ the set of terminal pairs $(s, t) \in I_{i,j}$ such that S_i intersects the (s, t)-path in T (note that then, for every $(s, t) \in I_{i,j} \setminus Q_{i,j}$, S_j intersects the (s, t)-path in T). Since the update procedure loops over every distribution, it considers at some point the distribution $(Q_{1,2}, Q_{1,3}, \ldots, Q_{p-1,p})$ and thus,

$$\mathsf{Tab}[v, O] \leq \sum_{i \in [p]} \mathcal{A}_v^*(u_i, \mathcal{P}_i^\pi)$$

where we set $\mathcal{P}_i^\pi = O_i \cup I_i \cup \bigcup_{i<j} Q_{i,j} \cup \bigcup_{i>j} I_{j,i} \setminus Q_{j,i}$ as above.

Now observe that for every $i \in [p]$ and every terminal pair $(s, t) \in \mathcal{P}|_{T_{u_i}} \cup O_i \cup I_i$, S_i must intersect the (s, t)-path in T, as $v \notin S$, and so

$$\mathtt{wt}(S_i) \geq \mathcal{A}_v^*(u_i, \mathcal{P}_i^\pi)$$

as $\mathcal{A}_v^*$ is correct. Therefore, $\mathsf{Tab}[v, O] \leq \mathtt{wt}(S)$ as claimed.

By the above, we conclude that for every set $O \subseteq O[v]$, $\mathsf{Tab}[v, O]$ is filled correctly. Finally, observe that for every $O \subseteq O[v]$,

$$|O| + \sum_{i \in [p-1]} \sum_{j > i} |I_{i,j}| \leq d$$

since v is d-light. Thus, for a fixed $O \subseteq O[v]$, there are

$$\prod_{i \in [p-1]} \prod_{j > i} 2^{|I_{i,j}|} = 2^{\sum_{i \in [p-1]} \sum_{j>i} |I_{i,j}|} \leq 2^{d - |O|}$$

distributions to consider in Step 2. It follows that $\mathsf{Tab}[v, O]$ can updated in time at most $2^{d-|O|} \cdot \mathcal{O}(\mathsf{T}(\mathcal{A}_v^{\mathsf{heavy}}))$ and thus, it takes at most

$$\sum_{i=0}^{d} \binom{d}{i} 2^{d-i} \cdot \mathcal{O}(\mathsf{T}(\mathcal{A}_v^{\mathsf{heavy}})) = 3^d \cdot \mathcal{O}(\mathsf{T}(\mathcal{A}_v^{\mathsf{heavy}}))$$

time to update $\mathsf{Tab}[v, \cdot]$. $\qquad\square$

Heavy Vertices with d-Light Parents. Let us now describe the subroutine $\mathcal{A}_v^{\mathsf{heavy}}$ for a d-light vertex v with at least one d-heavy child. Let u be a fixed d-heavy child of v. Denote by C_u the connected component of $T - \mathsf{light}(T, \mathcal{P}, d)$ containing u and by $N_u = N(C_u) \cap \mathsf{light}(T, \mathcal{P}, d) \setminus \{v\}$. If $N_u \neq \emptyset$, then we denote by $N_u = \{u_1, \ldots, u_\lambda\}$ for some $\lambda \leq \ell$, and for every $i \in [\lambda]$, we let p_i be the parent of u_i in T (note that, by definition, $p_i \in C_u$ for every $i \in [\lambda]$).

Given a set $Q \subseteq O[u]$, the basic idea is to "guess" for each u_i, a set O_i of pairs in $O[u_i]$ which are already cut by an optimal solution for the subtree T_{u_i}. We are then only interested in separating terminal pairs which intersect T_u and are not already separated by a solution in some T_{u_i}. By definition of the problem, we also do not have to separate the pairs in $O[u] \setminus Q$. By these observations, it suffices to only consider the subtree T_u' obtained from T_u after deleting all subtrees T_{u_i}. Because of this deletion, there might be pairs which do not start or end in T_u' but must be separated in T_u'. We take care of those by constructing a projection τ which maps the start and end point of the terminal pairs to the first vertex of the path which lies inside T_u'.

Remark 5.3. *The algorithm of Theorem 1.2 can, with an additional $\mathcal{O}(\log \mathbf{w})$-factor in the runtime, find the minimum weight of a $\mathcal{P}$-multicut in a tree with ℓ leaves, in time $2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n^{\mathcal{O}(1)}$. We denote this algorithm by $\mathcal{A}_{\mathsf{un\text{-}mc}}$.*

A *distribution* is a λ-tuple $(O_1, \ldots, O_\lambda)$ where for every $i \in [\lambda]$, $O_i \subseteq O[u_i]$. Given a set $Q \subseteq O[u]$, the algorithm $\mathcal{A}_v^{\mathsf{heavy}}$ proceeds as follows.

1. If $N_u = \emptyset$ then return the weight of the set

$$\mathcal{A}_{\mathsf{un\text{-}mc}}(T_u, \mathcal{P}|_{T_u} \cup \{(s, u) \mid (s, t) \in Q \text{ and } s \in V(T_u)\}, \mathtt{wt}|_{V(T_u)}).$$

2. Initialize $\mathsf{OPT} = \infty$.

3. Set $T_u' = T[V(T_u) \setminus \bigcup_{i \in [\lambda]} V(T_{u_i})]$.

4. Define the projection $\tau_u : V(T) \to V(T_u')$ where for all $v \in V(T)$

$$\tau_u(v) = \begin{cases} p_i & \text{if } v \in V(T_{u_i}), \\ u & \text{if } v \in V(T) \setminus V(T_u), \\ v & \text{otherwise.} \end{cases}$$

5. For every distribution $\pi = (O_1, \ldots, O_p)$ do:

    5.1. Let $\mathcal{P}_{u,\pi} = (Q \cup \mathcal{P}|_{T_u}) \setminus \bigcup_{i \in [\lambda]} (\mathcal{P}|_{T_{u_i}} \cup O_i)$.

    5.2. Set $\mathcal{P}_{u,\pi}' = \{(\tau_u(s), \tau_u(t)) \mid (s, t) \in \mathcal{P}_{u,\pi}\}$.

    5.3. Compute $M = \mathcal{A}_{\mathsf{un\text{-}mc}}(T_u', \mathcal{P}_{u,\pi}', \mathtt{wt}|_{V(T_u')})$

    5.4. Set $\mathsf{OPT} = \min\{\mathsf{OPT}, \mathtt{wt}(M) + \sum_{i \in [\lambda]} \mathsf{Tab}[u_i, O_i]\}$.

6. Return OPT.

Lemma 5.4. *For every d-light vertex v with at least one d-heavy child, $\mathcal{A}_v^{\mathsf{heavy}}$ is correct and runs in time $2^{d\ell} \cdot 2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n^{\mathcal{O}(1)}$.*

Proof. Let v be a d-light vertex in T with at least one d-heavy child and let u be one such child of v. Consider a set $Q \subseteq O[u]$. If $N_u = \emptyset$ then, denoting by $M = \mathcal{A}_{\mathsf{un\text{-}mc}}(T_u, \mathcal{P}|_{T_u} \cup \{(s, u) \mid (s, t) \in Q \text{ and } s \in V(T_u)\}, \mathtt{wt}|_{V(T_u)})$, it is clear that M is a subset of $V(T_u)$ of minimum weight such that M is a $\mathcal{P}|_{T_u}$-multicut and for every $(s, t) \in Q$, M intersects the (s, t)-path in T. Thus, $\mathcal{A}_v^{\mathsf{heavy}}$ indeed outputs the correct answer in this case.

Suppose next that $N_u = \{u_1, \ldots, u_\lambda\}$ for some $\lambda \in [\ell]$, and assume that for every $i \in [\lambda]$, $\mathsf{Tab}[u_i, \cdot]$ is correctly filled. Let $S \subseteq V(T_u)$ be a set of minimum weight such that S is a $\mathcal{P}|_{T_u}$-multicut and for every $(s, t) \in Q$, S intersects the (s, t)-path in T. For every $i \in [\lambda]$, let $S_i = S \cap V(T_{u_i})$ and let $O_i \subseteq O[u_i]$ be the set of terminal pairs (s, t) such that S_i intersects the (s, t)-path in T.

Since the algorithm loops over every distribution, it considers at some point the distribution $\pi = (O_1, \ldots, O_\lambda)$: let $M^* = \mathcal{A}_{\mathsf{un\text{-}mc}}(T_u', \mathcal{P}_{u,\pi}', \mathtt{wt}|_{V(T_u')})$ where $\mathcal{P}_{u,\pi}'$ and T_u' are as defined in the algorithm. We aim to show that $\mathtt{wt}(S) = \mathtt{wt}(M^*) + \sum_{i \in [\lambda]} \mathsf{Tab}[u_i, O_i]$.

To this end, let us show that $S^* = S \setminus \bigcup_{i \in [\lambda]} S_i$ is a $\mathcal{P}_{u,\pi}'$-multicut in T_u'. Consider a terminal pair $(s, t) \in \mathcal{P}_{u,\pi}'$. If $(s, t) \in \mathcal{P}_{u,\pi} \cap \mathcal{P}_{u,\pi}'$, then S^* intersects the (s, t)-path in T_u' since S is a $\mathcal{P}|_{T_u}$-multicut and $s, t \in V(T_u) \setminus \bigcup_{i \in [\lambda]} V(T_{u_i})$.

Suppose next that $(s, t) \in \mathcal{P}_{u,\pi}' \setminus \mathcal{P}_{u,\pi}$. Then (s, t) corresponds to a terminal pair $(a, b) \in \mathcal{P}_{u,\pi} \setminus \mathcal{P}_{u,\pi}'$ such that $\tau_u(a) = s$ and $\tau_u(b) = t$. Since by construction, S intersects the (a, b)-path in T and for every $i \in [\lambda]$, $(a, b) \notin O_i$, it follows that S^* intersects the (a, b)-path in T and, a fortiori, the (s, t)-path in T_u'.

Therefore, S^* is a $\mathcal{P}_{u,\pi}'$-multicut in T_u' as claimed; in particular, $\mathtt{wt}(M^*) \leq \mathtt{wt}(S^*)$ by minimality of M^*. Now observe that for every $i \in [\lambda]$, S_i is a $\mathcal{P}|_{T_{u_i}}$-multicut such that for every

19

$(s,t) \in O_i$, S_i intersects the (s,t)-path in T, and thus, $\mathsf{wt}(S_i) \geq \mathsf{Tab}[u_i, O_i]$ as $\mathsf{Tab}[u_i, \cdot]$ is correctly filled by assumption. It follows that $\mathsf{wt}(M^*) + \sum_{i \in [\lambda]} \mathsf{Tab}[u_i, O_i] \leq \mathsf{wt}(S)$: indeed, since $\mathsf{wt}(M^*) \leq \mathsf{wt}(S^*)$ and $\mathsf{wt}(S^*) = \mathsf{wt}(S) - \sum_{i \in [\lambda]} \mathsf{wt}(S_i)$,

$$\mathsf{wt}(M^*) \leq \mathsf{wt}(S) - \sum_{i \in [\lambda]} \mathsf{wt}(S_i) \leq \mathsf{wt}(S) - \sum_{i \in [\lambda]} \mathsf{Tab}[u_i, O_i].$$

To prove the converse inequality, for every $i \in [\lambda]$, let S_i^* be a set of minimum weight such that S_i^* is a $\mathcal{P}|_{T_{u_i}}$-multicut and for every $(s,t) \in O_i$, S_i^* intersects the (s,t)-path in T. We contend that $M = M^* \cup \bigcup_{i \in [\lambda]} S_i^*$ is a $\mathcal{P}|_{T_u}$-multicut such that for every $(s,t) \in Q$, M intersects the (s,t)-path in T.

Indeed, consider a terminal pair $(s,t) \in \mathcal{P}|_{T_u} \cup Q$. If $(s,t) \in \mathcal{P}|_{T_{u_i}}$ for some $i \in [\lambda]$, then M intersects the (s,t)-path in T since S_i^* is a $\mathcal{P}|_{T_{u_i}}$-multicut by definition. Similarly, if $(s,t) \in O_i$ for some $i \in [\lambda]$, then M intersects the (s,t)-path in T since S_i^* intersects this path by definition. Thus, let us assume that $(s,t) \in \mathcal{P}_{u,\pi}$. Suppose first that $\{s,t\} \cap V(T_{u_i}) \neq \emptyset$ for some $i \in [\lambda]$, say $s \in V(T_{u_i})$ without loss of generality. If $t \in V(T_{u_j})$ for some $j \in [\lambda]$, then $(p_i, p_j) \in \mathcal{P}'_{u,\pi}$ and so, M intersects the (s,t)-path in T since M^* intersects the (p_i, p_j)-path in T'_u by definition. If $t \in V(T_u) \setminus \bigcup_{j \in [\lambda]} V(T_{u_j})$, then $(p_i, t) \in \mathcal{P}'_{u,\pi}$ and so, M intersects the (s,t)-path in T since M^* intersects the (p_i, t)-path in T'_u by definition. Finally, if $t \in V(T) \setminus V(T_u)$, then $(p_i, u) \in \mathcal{P}'_{u,\pi}$ and so, M intersects the (s,t)-path in T since M^* intersects the (p_i, u)-path in T'_u by definition.

Second, suppose that $\{s,t\} \cap V(T_{u_i}) = \emptyset$ for every $i \in [\lambda]$. If $s,t \in V(T_u)$, then M intersects the (s,t)-path in T since M^* intersects this path by definition. Otherwise, exactly one of s and t belongs to $V(T_u)$, say $s \in V(T_u)$ without loss of generality, in which case $(s,u) \in \mathcal{P}'_{u,\pi}$ and so, M intersects the (s,t)-path in T since M^* intersects the (s,u)-path in T'_u by definition. Thus, M is as claimed.

By minimality of S, it follows that $\mathsf{wt}(M) \geq \mathsf{wt}(S)$; but $\mathsf{wt}(M) = \mathsf{wt}(M^*) + \sum_{i \in [\lambda]} \mathsf{Tab}[u_i, O_i]$, as for every $i \in [\lambda]$, $\mathsf{Tab}[u_i, \cdot]$ is correctly filled, and thus, the converse inequality holds as well. Combined with the above, we conclude that the algorithm $\mathcal{A}_v^{\mathsf{heavy}}$ is correct.

Finally, let us argue that the algorithm $\mathcal{A}_v^{\mathsf{heavy}}$ runs in $2^{d\ell} \cdot 2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n^{\mathcal{O}(1)}$-time. First, Step 1. can be done in $2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n^{\mathcal{O}(1)}$-time by Theorem 1.2 since, in this case, T_u has at most ℓ leaves. Otherwise, observe that for a fixed distribution, the most computationally demanding step is the call to $\mathcal{A}_{\mathsf{un-mc}}$ in Step 5.3., which takes $2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n^{\mathcal{O}(1)}$-time by Theorem 1.2, as T'_u has at most ℓ leaves. Now note that there are at most $2^{d\ell}$ distributions to consider: indeed, $|N_u| \leq \ell$ and for every $i \in [\lambda]$, $|2^{O[u_i]}| \leq 2^d$ as u_i is d-light. Thus, Step 5. takes $2^{d\ell} \cdot 2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n^{\mathcal{O}(1)}$-time in total and so, the algorithm indeed runs in the stated time. $\qquad \square$

6 Future Directions

The natural question to ask is, whether the running times of our algorithms can be improved. Faster algorithms for the arcless instances in [15], directly yield faster algorithms for UWMC-TREE parameterized by the number of leaves. Another interesting question is to determine the parameterized complexity of the (bi-objective) wMC problem with respect to structural parameters such as the number of leaves. There it seems difficult to use the flow augmentation technique from [15], since such a step takes exponential time in the solution size, but the number of the leaves in the input may be much smaller than the solution size.

Another interesting follow up question is to determine if one can use the directed flow augmentation to resolve the parameterized complexity of WEIGHTED STEINER MULTICUT on trees, where given a tree T, sets $P_1, \ldots, P_r \subseteq V(T)$ each of size $p \geq 1$, a weight function $\mathsf{wt} : V(T) \to \mathbb{N}$ and positive integers $\mathbf{w}, k$, the goal is to determine if there exists a set $S \subseteq V(T)$ such that $|S| \leq k$, $\mathsf{wt}(S) \leq \mathbf{w}$ and for each $i \in [q]$ there exists $u_i, v_i \in P_i$ such that $T - S$ has no (u_i, v_i)-path. Observe that WMC-TREE is a special case of this problem when $p = 2$.

References

[1] Nicolas Bousquet, Jean Daligault, Stéphan Thomassé, and Anders Yeo. A polynomial kernel for multicut in trees. In Susanne Albers and Jean-Yves Marion, editors, *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, volume 3 of *LIPIcs*, pages 183–194. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2009. doi: 10.4230/LIPIcs.STACS.2009.1824. URL https://doi.org/10.4230/LIPIcs.STACS.2009.1824.

[2] Nicolas Bousquet, Jean Daligault, and Stéphan Thomassé. Multicut is FPT. *SIAM J. Comput.*, 47(1):166–207, 2018. doi: 10.1137/140961808. URL https://doi.org/10.1137/140961808.

[3] Jianer Chen, Jia-Hao Fan, Iyad Kanj, Yang Liu, and Fenghui Zhang. Multicut in trees viewed through the eyes of vertex cover. *Journal of Computer and System Sciences*, 78(5):1637–1650, 2012.

[4] Rajesh Chitnis, László Egri, and Dániel Marx. List H-coloring a graph by removing few vertices. *Algorithmica*, 78(1):110–146, 2017. doi: 10.1007/s00453-016-0139-6. URL https://doi.org/10.1007/s00453-016-0139-6.

[5] Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. *SIAM J. Comput.*, 42(4):1674–1696, 2013. doi: 10.1137/12086217X. URL https://doi.org/10.1137/12086217X.

[6] Rajesh Hemant Chitnis, Marek Cygan, Mohammad Taghi Hajiaghayi, and Dániel Marx. Directed subset feedback vertex set is fixed-parameter tractable. *ACM Trans. Algorithms*, 11(4):28:1–28:28, 2015. doi: 10.1145/2700209. URL https://doi.org/10.1145/2700209.

[7] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. ISBN 978-3-319-21274-6. doi: 10.1007/978-3-319-21275-3. URL https://doi.org/10.1007/978-3-319-21275-3.

[8] Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23(4):864–894, 1994. doi: 10.1137/S0097539792225297. URL https://doi.org/10.1137/S0097539792225297.

[9] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012. ISBN 978-3-642-14278-9.

[10] M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified np-complete problems. In Robert L. Constable, Robert W. Ritchie, Jack W. Carlyle, and Michael A. Harrison, editors, *Proceedings of the 6th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1974, Seattle, Washington, USA*, pages 47–63. ACM, 1974. doi: 10.1145/800119.803884. URL https://doi.org/10.1145/800119.803884.

[11] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997. doi: 10.1007/BF02523685. URL https://doi.org/10.1007/BF02523685.

[12] Jiong Guo and Rolf Niedermeier. Fixed-parameter tractability and data reduction for multicut in trees. *Networks*, 46(3):124–135, 2005. doi: 10.1002/net.20081. URL https://doi.org/10.1002/net.20081.

[13] Jiong Guo and Rolf Niedermeier. Exact algorithms and applications for tree-like weighted set cover. *J. Discrete Algorithms*, 4(4):608–622, 2006. doi: 10.1016/j.jda.2005.07.005. URL https://doi.org/10.1016/j.jda.2005.07.005.

[14] Iyad Kanj, Guohui Lin, Tian Liu, Weitian Tong, Ge Xia, Jinhui Xu, Boting Yang, Fenghui Zhang, Peng Zhang, and Binhai Zhu. Algorithms for cut problems on trees. In *International Conference on Combinatorial Optimization and Applications*, pages 283–298. Springer, 2014.

[15] Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Directed flow-augmentation. To appear in Proceedings of the 54th Annual ACM Symposium on Theory of Computing (STOC 2022). Full version at https://arxiv.org/abs/2111.03450.

[16] Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Solving hard cut problems via flow-augmentation. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 149–168. SIAM, 2021. doi: 10.1137/1.9781611976465.11. URL https://doi.org/10.1137/1.9781611976465.11.

[17] Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. *J. ACM*, 67(3):16:1–16:50, 2020. doi: 10.1145/3390887. URL https://doi.org/10.1145/3390887.

[18] Stefan Kratsch, Marcin Pilipczuk, Michal Pilipczuk, and Magnus Wahlström. Fixed-parameter tractability of multicut in directed acyclic graphs. *SIAM J. Discret. Math.*, 29 (1):122–144, 2015. doi: 10.1137/120904202. URL https://doi.org/10.1137/120904202.

[19] Daniel Lokshtanov and Dániel Marx. Clustering with local restrictions. *Inf. Comput.*, 222:278–292, 2013. doi: 10.1016/j.ic.2012.10.016. URL https://doi.org/10.1016/j.ic.2012.10.016.

[20] Daniel Lokshtanov and M. S. Ramanujan. Parameterized tractability of multiway cut with parity constraints. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, volume 7391 of *Lecture Notes in Computer Science*, pages 750–761. Springer, 2012. doi: 10.1007/978-3-642-31594-7_63. URL https://doi.org/10.1007/978-3-642-31594-7_63.

[21] Dániel Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006. doi: 10.1016/j.tcs.2005.10.007. URL https://doi.org/10.1016/j.tcs.2005.10.007.

[22] Dániel Marx and Igor Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. *SIAM J. Comput.*, 43(2):355–388, 2014. doi: 10.1137/110855247. URL https://doi.org/10.1137/110855247.

[23] Mihalis Yannakakis, Paris C. Kanellakis, Stavros S. Cosmadakis, and Christos H. Papadimitriou. Cutting and partitioning a graph aifter a fixed pattern (extended abstract). In Josep Díaz, editor, *Automata, Languages and Programming, 10th Colloquium, Barcelona, Spain, July 18-22, 1983, Proceedings*, volume 154 of *Lecture Notes in Computer Science*, pages 712–722. Springer, 1983. doi: 10.1007/BFb0036950. URL https://doi.org/10.1007/BFb0036950.