Mathematical Programming
https://doi.org/10.1007/s10107-022-01842-3

FULL LENGTH PAPER

Series B

Fixed parameter approximation scheme for min-max k-cut

Karthekeyan Chandrasekaran[1] · Weihang Wang[1]

Received: 27 May 2021 / Accepted: 30 May 2022

Abstract

We consider the graph k-partitioning problem under the min-max objective, termed as MINMAX k-CUT. The input here is a graph $G = (V, E)$ with non-negative integral edge weights $w : E \rightarrow \mathbb{Z}_+$ and an integer $k \geq 2$ and the goal is to partition the vertices into k non-empty parts $V_1, \ldots, V_k$ so as to minimize $\max_{i=1}^{k} w(\delta(V_i))$. Although minimizing the sum objective $\sum_{i=1}^{k} w(\delta(V_i))$, termed as MINSUM k-CUT, has been studied extensively in the literature, very little is known about minimizing the max objective. We initiate the study of MINMAX k-CUT by showing that it is NP-hard and W[1]-hard when parameterized by k, and design a parameterized approximation scheme when parameterized by k. The main ingredient of our parameterized approximation scheme is an exact algorithm for MINMAX k-CUT that runs in time $(\lambda k)^{O(k^2)} n^{O(1)} + O(m)$, where λ is value of the optimum, n is the number of vertices, and m is the number of edges. Our algorithmic technique builds on the technique of Lokshtanov, Saurabh, and Surianarayanan (FOCS, 2020) who showed a similar result for MINSUM k-CUT. Our algorithmic techniques are more general and can be used to obtain parameterized approximation schemes for minimizing ℓ_p-norm measures of k-partitioning for every $p \geq 1$.

Keywords k-cut · Min-max objective · Parameterized approximation scheme

Mathematics Subject Classification 05C85 · 68W25

An extended abstract of this work appeared in the 22nd conference on Integer Programming and Combinatorial Optimization (IPCO 2021).

✉ Weihang Wang
weihang3@illinois.edu

Karthekeyan Chandrasekaran
karthe@illinois.edu

1 University of Illinois Urbana-Champaign, Champaign, USA

1 Introduction

Graph partitioning problems are fundamental for their intrinsic theoretical value as well as applications in clustering. In this work, we consider graph partitioning under the *minmax* objective. The input here is a graph $G = (V, E)$ with non-negative integral edge weights $w : E \to \mathbb{Z}_+$ along with an integer $k \geq 2$ and the goal is to partition the vertices of G into k non-empty parts $V_1, V_2, \ldots, V_k$ so as to minimize $\max_{i=1}^{k} w(\delta(V_i))$; here, $\delta(V_i)$ is the set of edges which have exactly one end-vertex in V_i and $w(\delta(V_i)) := \sum_{e\in\delta(V_i)} w(e)$ is the total weight of the edges in $\delta(V_i)$. We refer to this problem as MINMAX k-CUT.
Motivations. Minmax objective for optimization problems has an extensive literature in approximation algorithms. It is relevant in scenarios where the goal is to achieve fairness/balance—e.g., load balancing in multiprocessor scheduling, discrepancy minimization, min-degree spanning tree, etc. In the context of graph cuts and partitioning, recent works (e.g., see [1, 5, 17]) have proposed and studied alternative minmax objectives that are different from MINMAX k-CUT.

The complexity of MINMAX k-CUT was also raised as an open problem by Lawler [21]. Given a partition $V_1, \ldots, V_k$ of the vertex set of an input graph, one can measure the quality of the partition in various natural ways. Two natural measures are (i) the max objective given by $\max_{i=1}^{k} w(\delta(V_i))$ and (ii) the sum objective given by $\sum_{i=1}^{k} w(\delta(V_i))$. We will discuss other *ℓ_p-norm measures* later. Once a measure is defined, a corresponding optimization problem involves finding a partition that minimizes the measure. We will denote the optimization problem where the goal is to minimize the sum objective as MINSUM k-CUT.

MINSUM k-CUT *and prior works.* For $k = 2$, the objectives in MINMAX k-CUT and MINSUM k-CUT coincide owing to the symmetric nature of the graph cut function (i.e., $w(\delta(S)) = w(\delta(V \setminus S))$ for all $S \subseteq V$) but the objectives differ for $k \geq 3$. MINSUM k-CUT has been studied extensively in the algorithms community leading to fundamental graph structural results. We briefly recall the literature on MINSUM k-CUT.

Goldschmidt and Hochbaum [11, 12] showed that MINSUM k-CUT is NP-hard when k is part of input by a reduction from CLIQUE and designed the first polynomial time algorithm for fixed k. Their algorithm runs in time $n^{O(k^2)}$, where n is the number of vertices in the input graph. Subsequently, Karger and Stein [19] gave a random contraction based algorithm that runs in time $\tilde{O}(n^{2k-2})$. Thorup [30] gave a tree-packing based deterministic algorithm that runs in time $\tilde{O}(n^{2k})$. The last couple of years has seen renewed interests in MINSUM k-CUT with exciting progress [6, 8, 13–16, 22, 24]. Very recently, Gupta, Harris, Lee, and Li [13, 16] have shown that the Karger-Stein algorithm in fact runs in $\tilde{O}(n^k)$ time; $n^{(1-o(1))k}$ seems to be a lower bound on the run-time of any algorithm [22]. The hardness result of Goldschmidt and Hochbaum as well as their algorithm inspired Saran and Vazirani [27] to consider MINSUM k-CUT when k is part of input from the perspective of approximation. They showed the first polynomial-time 2-approximation for MINSUM k-CUT. Alternative 2-approximations have also been designed subsequently [25, 26, 31]. For k being a part of the input, Manurangsi [24] showed that there does not exist a polynomial-time $(2-\epsilon)$-approximation for any constant $\epsilon > 0$ under the Small Set Expansion Hypothesis.

Minsum k-cut has also been investigated from the perspective of fixed-parameter algorithms. It is known that Minsum k-cut when parameterized by k is W[1]-hard and does not admit a $f(k)n^{o(1)}$-time algorithm for any function $f(k)$ [7, 10]. Motivated by this hardness result and Manurangsi's $(2-\epsilon)$-inapproximability result, Gupta, Lee, and Li [14] raised the question of whether there exists a *parameterized approximation algorithm* for Minsum k-cut when parameterized by k, i.e., can one obtain a $(2-\epsilon)$-approximation in time $f(k)n^{O(1)}$ for some constant $\epsilon > 0$? As a proof of concept, they designed a 1.9997-approximation algorithm that runs in time $2^{O(k^6)}n^{O(1)}$ [14] and a $(1+\epsilon)$-approximation algorithm that runs in time $(k/\epsilon)^{O(k)}n^{k+O(1)}$ [15]. Subsequently, Kawarabayashi and Lin [20] designed a $(5/3+\epsilon)$-approximation algorithm that runs in time $2^{O(k^2\log k)}n^{O(1)}$. This line of work culminated in a *parameterized approximation scheme* when parameterized by k—Lokshtanov, Saurabh, and Surianarayanan [23] designed a $(1+\epsilon)$-approximation algorithm that runs in time $(k/\epsilon)^{O(k)}n^{O(1)}$. We emphasize that, from the perspective of algorithm design, a parameterized approximation scheme is more powerful than a parameterized approximation algorithm.

Fixed-terminal variants. A natural approach to solve both Minmax k-cut and Minsum k-cut is to solve their fixed-terminal variants: The input here is a graph $G=(V,E)$ with non-negative integral edge costs $w: E \to \mathbb{Z}_+$ along with k terminals $v_1,\ldots,v_k \in V$ and the goal is to partition the vertices into k parts $V_1,\ldots,V_k$ such that $v_i \in V_i$ for every $i \in [k]$ so as to minimize the measure of interest for the partition. The fixed-terminal variant of Minsum k-cut, popularly known as Multiway cut, is NP-hard for $k \geq 3$ [9] and has a rich literature. It admits a 1.2965-approximation [28] and does not admit a $(1.20016-\epsilon)$-approximation for any constant $\epsilon > 0$ under the unique games conjecture [3]. The fixed-terminal variant of Minmax k-cut, known as Minmax Multiway cut, is NP-hard for $k \geq 4$ [29] and admits an $O(\sqrt{\log n \log k})$-approximation [2]. Although fixed-terminal variants are natural approaches to solve global cut problems (similar to using min $\{s,t\}$-cut to solve global min-cut), they have two limitations: (1) they are not helpful when k is part of input and (2) even for fixed k, they do not give the best algorithms (e.g., even for $k=3$, Multiway cut is NP-hard while Minsum k-cut is solvable in polynomial time as discussed above).

Minmax k-cut *vs* Minsum k-cut. There are fundamental structural differences between Minmax k-cut and Minsum k-cut. The optimal solution to Minsum k-cut satisfies certain nice properties: (i) If the input graph is connected, then every part in an optimal partition for Minsum k-cut induces a connected subgraph. (ii) If the input graph is disconnected, then there exists an optiumal partition for Minsum k-cut such that every part in is completely contained within a connected component. Hence, Minsum k-cut is also phrased as the problem of deleting a subset of edges with minimum weight so that the resulting graph contains at least k connected components. However, these nice properties fail to hold for Minmax k-cut as illustrated by examples in Figs. 1 and 2.

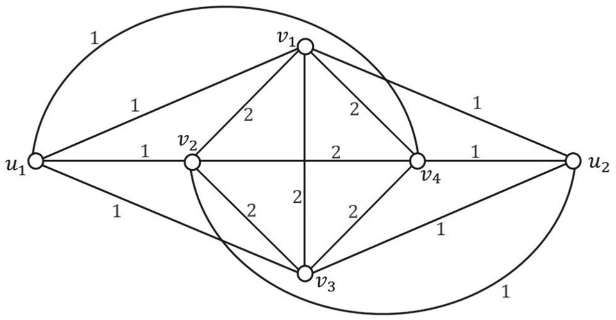

Fig. 1 An example where the unique optimum partition for MINMAX k-CUT for $k = 5$ induces a disconnected part. The edge weights are as shown. Every 5-partition in this example necessarily consists of one part with 2 vertices and four singleton parts. If the part with 2 vertices is $\{u_1, u_2\}$, then the objective value is 8. If the part with 2 vertices is $\{u_i, v_j\}$ where $i \in [2]$ and $j \in [4]$, then the objective value is 10. If the part with 2 vertices is $\{v_i, v_j\}$ where $i, j \in [4]$, then the objective value is 12. Hence the optimum partition for minmax 5-cut is $(\{u_1, u_2\}, \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\})$, where the first part induces a disconnected subgraph

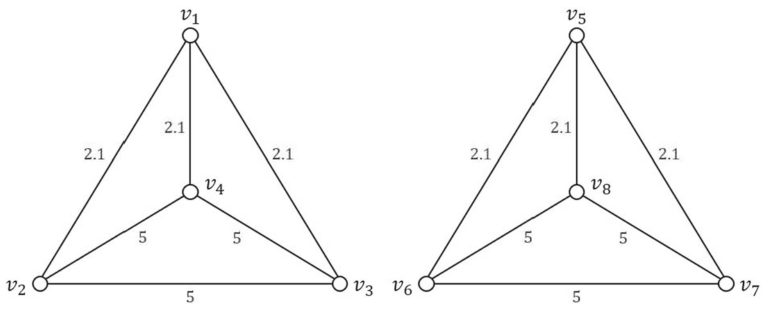

Fig. 2 An example of a disconnected graph where the unique optimum partition for MINMAX k-CUT for $k = 7$ has a part that intersects two connected components. The edge weights are as shown. The unique optimum k-partition for $k = 7$ is $(\{v_1, v_5\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_6\}, \{v_7\}, \{v_8\})$, where the first part intersects two different components components

MINMAX k-CUT *for fixed* k. For fixed k, there is an easy approach to solve MINMAX k-CUT based on the following observation: For a given instance, an optimum solution to MINMAX k-CUT is a k-approximate optimum to MINSUM k-CUT. [1] The randomized algorithm of Karger and Stein implies that the number of k-approximate solutions to MINSUM k-CUT is $n^{O(k^2)}$ and they can all be enumerated in polynomial time [13, 16, 19] (also see [6]). These two facts immediately imply that MINMAX k-CUT can be solved in $n^{O(k^2)}$ time. We recall that the graph cut function is symmetric and submodular.[2] In a recent work, Chandrasekaran and Chekuri [4] show that the more general problem of

[1] If $(V_1, \ldots, V_k)$ is an optimum k-partition for MINMAX k-CUT with optimum value OPT_{mm} and OPT_{ms} is the optimum value for MINSUM k-CUT, then we have that $\sum_{i=1}^{k} w(\delta(V_i)) \le k\text{OPT}_{mm} \le k\text{OPT}_{ms}$.

[2] A function $f : 2^V \to \mathbb{R}$ is symmetric if $f(S) = f(V \setminus S)$ for all $S \subseteq V$ and is submodular if $f(A) + f(B) \ge f(A \cap B) + f(A \cup B)$.

min-max symmetric submodular k-partition[3] is also solvable in time $n^{O(k^2)}T$, where n is the size of the ground set and T is the time to evaluate the input submodular function on a given set.

1.1 Results

In this work, we focus on MINMAX k-CUT when k is part of input. We first show that MINMAX k-CUT is strongly NP-hard. Our reduction also implies that it is W[1]-hard when parameterized by k, i.e., there does not exist a $f(k)n^{O(1)}$-time algorithm for any function $f(k)$.

Theorem 1 MINMAX k-CUT *is strongly NP-hard and W[1]-hard when parameterized by k.*

Our hardness reduction also implies that MINMAX k-CUT does not admit an algorithm that runs in time $n^{o(k)}$ assuming the exponential time hypothesis. Given the hardness result, it is natural to consider approximations and fixed-parameter tractability. Using the known 2-approximation for MINSUM k-CUT and the observation that the optimum value of MINSUM k-CUT is at most k times the optimum value of MINMAX k-CUT, it is easy to get a $(2k)$-approximation for MINMAX k-CUT. An interesting open question is whether we can improve the approximability/inapproximability.

The hardness results also raise the question of whether MINMAX k-CUT admits a parameterized approximation algorithm when parameterized by k or, going a step further, does it admit a parameterized approximation scheme when parameterized by k? We resolve this question affirmatively by designing a parameterized approximation scheme. Let $G = (V, E)$ be a graph with non-negative integral edge weights $w : E \to \mathbb{Z}_+$. We write G_w to denote the graph with edge weights w. For a partition $(V_1, \ldots, V_k)$ of V, we define

$$\text{cost}_{G_w}(V_1, \ldots, V_k) := \max\{w(\delta(V_i)) : i \in [k]\}.$$

We will denote the minimum cost of a k-partition in G_w by $\text{OPT}(G_w, k)$. We will call an instance to be weighted if G has no parallel edges (with edge weights being arbitrary) and an instance to be unweighted if all edge weights are unit and the graph G possibly has parallel edges. The following is our algorithmic result showing that MINMAX k-CUT admits a parameterized approximation scheme when parameterized by k.

Theorem 2 *There exists a randomized algorithm that takes as input a weighted instance of* MINMAX k-CUT, *namely an n-vertex simple graph $G = (V, E)$ with edge weights $w : E \to \mathbb{Z}_+$ and an integer $k \geq 2$, along with an $\epsilon \in (0, 1)$, and runs in time $(k/\epsilon)^{O(k^2)} n^{O(1)} \log^2(\sum_{e \in E} w(e))$ to return a partition $\mathcal{P}$ of the vertices of G such that $\text{cost}_{G_w}(\mathcal{P}) \leq (1 + \epsilon)OPT(G_w, k)$ with high probability.*

[3] In the min-max symmetric submodular k-partition problem, the input is a symmetric submodular function $f : 2^V \to \mathbb{R}$ given by an evaluation oracle, and the goal is to partition the ground set V into k non-empty parts $V_1, \ldots, V_k$ so as to minimize $\max_{i=1}^{k} f(V_i)$.

We note that $\log(\sum_{e\in E} w(e))$ is polynomial in the size of the input. Throughout, high probability refers to $1 - o(1/n)$. Theorem 2 can be viewed as the counterpart of the parameterized approximation scheme for MINSUM k-CUT due to Lokshtanov, Saurabh, and Surianarayanan [23] but for MINMAX k-CUT. The central component of our parameterized-approximation scheme given in Theorem 2 is the following result which shows a fixed-parameter algorithm for MINMAX k-CUT in unweighted instances when parameterized by k and the solution size. For an unweighted graph $G = (V, E)$, we define the cost of a partition $(V_1, \ldots, V_k)$ of V as $\text{cost}_G(V_1, \ldots, V_k) := \max\{|\delta(V_i)| : i \in [k]\}$.

Theorem 3 *There exists an algorithm that takes as input an unweighted instance $G = (V, E)$ of* MINMAX k-CUT, *namely an n-vertex m-edge graph $G = (V, E)$ and an integer $k \geq 2$, along with an integer λ and runs in time $(k\lambda)^{O(k^2)} n^{O(1)} + O(km)$ to determine if there exists a k-partition $(V_1, \ldots, V_k)$ of V such that $\text{cost}_G(V_1, \ldots, V_k) \leq \lambda$ and if so, returns an optimum partition for* MINMAX k-CUT *on G.*

We emphasize that the algorithm in Theorem 3 is deterministic.

1.2 Outline of techniques

Our NP-hardness and W[1]-hardness results for MINMAX k-CUT are based on a reduction from the clique problem. Our reduction is an adaptation of the reduction from the clique problem to MINSUM k-CUT due to Downey et al [10].

Our randomized algorithm for Theorem 2 essentially reduces the input weighted instance of MINMAX k-CUT to an instance where Theorem 3 can be applied: we reduce the instance to an unweighted instance with optimum value $O(k/\epsilon^3)\log n$, i.e., the optimum value is logarithmic in the number of vertices, and with $(n/\epsilon)^{O(1)}$ edges. Moreover, the reduction runs in time $(n/\epsilon)^{O(1)} \log^2(\sum_{e\in E} w_e)$. Applying Theorem 3 to the reduced instance yields a run-time of

$$\begin{aligned}\left(\left(\frac{k^2}{\epsilon^3}\right)\log n\right)^{O(k^2)} n^{O(1)} &= \left(\frac{k}{\epsilon}\right)^{O(k^2)} (\log n)^{O(k^2)} n^{O(1)} \\ &= \left(\frac{k}{\epsilon}\right)^{O(k^2)} (k^{O(k^2)} + n) n^{O(1)} \\ &= \left(\frac{k}{\epsilon}\right)^{O(k^2)} n^{O(1)}.\end{aligned}$$

Hence, the total run-time (including the reduction time) is $(k/\epsilon)^{O(k^2)} n^{O(1)} \log^2 (\sum_{e\in E} w_e)$, thereby proving Theorem 2.

We now briefly describe the reduction to an unweighted instance with logarithmic optimum: (i) Firstly, we do a standard knapsack PTAS-style rounding procedure to convert the instance to an unweighted instance with a $(1 + \epsilon)$-factor loss (see Lemma 15). (ii) Secondly, we delete cuts with small value to ensure that all connected components in the graph have large min-cut value, i.e., have min-cut value at least

$\epsilon \mathrm{OPT}/k$—this deletion procedure can remove at most ϵOPT edges and hence, a $(1+\epsilon)$-approximate solution in the resulting graph gives a $(1+\mathsf{O}(\epsilon))$-approximate solution in the original graph. (iii) Finally, we do a random sampling of edges with probability $p := \Theta(k \log n/(\epsilon^3 \mathrm{OPT}))$. This gives a subgraph that preserves all cut values within a $(1 \pm \epsilon)$-factor when scaled by p with high probability. The preservation of all cut values also implies that the optimum value to Minmax k-cut is also preserved within a $(1 \pm \epsilon)$-factor. The scaling factor of p allows us to conclude that the optimum in the subsampled graph is $\mathsf{O}((k/\epsilon^3)) \log n$. We note that this three step reduction follows the same ideas as that of [23] who designed a parameterized approximation scheme for Minsum k-cut. Our contribution to the reduction is simply showing that their reduction ideas also apply to Minmax k-cut (see Sect. 4 for details).

The main contribution of our work is in proving Theorem 3, i.e., giving a fixed-parameter algorithm for Minmax k-cut when parameterized by k and the solution size. We discuss this now. Our algorithm is designed for the case of connected graphs; we handle a disconnected graph by replacing every edge with multiple copies and using additional edges to make the graph connected—this does not change the optimum (see proof of Theorem 3). At a high-level, we exploit the tools developed by [23] who designed a dynamic program based fixed-parameter algorithm for Minsum k-cut when parameterized by k and the solution size. Our algorithm for Minmax k-cut is also based on a dynamic program. However, since we are interested in Minmax k-cut, the subproblems in our dynamic program are completely different from that of [23]. We begin with the observation that an optimum solution to Minmax k-cut is a k-approximate optimum to Minsum k-cut. Chekuri, Quanrud, and Xu [6] showed that given a graph G, there exists a polynomial-time algorithm to find a polynomial-sized family of spanning trees such that for any k-partition Π that is an α-approximate partition for Minsumk−cut, there exists a spanning tree T in the family such that the number of edges of T crossing Π is at most $\mathsf{O}(\alpha k)$. This result coupled with the observation that an optimum solution to Minmax k-cut is a k-approximate optimum to Minsum k-cut allows us to obtain, in polynomial time, a polynomial-sized family of spanning trees such that the number of edges of one of the spanning trees T in the family has $\mathsf{O}(k^2)$ edges crossing an optimum k-partition to Minmax k-cut optimum. Let us fix such a spanning tree T (our algorithm would iterate over all spanning trees in the returned family). We will call a partition Π with $\mathsf{O}(k^2)$ edges of the spanning tree T crossing Π to be a T-feasible partition. Next, we use the tools of [23] to generate, in polynomial time, a suitable tree decomposition of the input graph—let us call this a *good* tree decomposition. The central intuition underlying our algorithm is to use the spanning tree T to guide a dynamic program on the good tree decomposition.

As mentioned before, our dynamic program is different from that of [23]. We now sketch the details of our dynamic program. For simplicity, we assume that we have a value $\lambda \geq \mathrm{OPT}(G,k)$. The *adhesion* of a tree node t in a tree decomposition, denoted A_t, is the intersection of the bag corresponding to t with that of its parent (the adhesion of the root node of the tree decomposition is the empty set). The good tree decomposition that we generate has low adhesion, i.e., the adhesion size is $\mathsf{O}(\lambda k)$ for every tree node. In order to define our sub-problems for a tree node t, we consider the set $\mathcal{F}^{A_t}$ of all possible partitions $\mathcal{P}_{A_t}$ of the adhesion A_t containing at most k parts such that $\mathcal{P}_{A_t}$ can be extended into a partition of the entire vertex set that is T-feasible. A simple counting argument shows that $|\mathcal{F}_{A_t}| = (\lambda k)^{O(k^2)}$ (see Lemma 4). Now consider a Boolean function $f_t : \mathcal{F}^{A_t} \times \{0, 1, \ldots, \lambda\}^k \times \{0, 1, \ldots, 2k^2\} \to \{0, 1\}$. We note that the domain of the function is small, i.e., $(\lambda k)^{O(k^2)}$. Let $(\mathcal{P}_{A_t}, \bar{x}, d)$ denote an argument to the function. The function aims to determine if there exists a partition $\mathcal{P}$ of the union of the bags descending from t in the tree decomposition (call this set of vertices to be V_t) so that (i) the projection of the partition $\mathcal{P}$ to A_t is exactly $\mathcal{P}_{A_t}$, (ii) the number of edges with exactly one end-vertex in the i^{th} part of $\mathcal{P}$ in the subgraph $G[V_t]$ is exactly x_i for all $i \in [k]$, and (iii) the number of tree edges crossing the partition $\mathcal{P}$ is at most d. If we can compute such a function f_r for the root node r of the tree decomposition, then it records the cut values of all partitions satisfying the three conditions with at most k parts and can be used to find the optimum value of MINMAX k-CUT, namely $\text{OPT}(G, k)$.

However, we are unable to solve the sub-problem (i.e., compute such a function f_t) based on the sub-problem values of the children of t. We observe that instead of solving this sub-problem exactly, a weaker goal of finding a function that satisfies a certain f-correct and f-sound properties suffices (see Definition 8 for these properties and Proposition 1). We show that this weaker goal of computing an f-correct and f-sound function f_t based on f-correct and f-sound functions $f_{t'}$ for all children t' of t can be achieved in time $(\lambda k)^{O(k^2)} n^{O(1)} + O(m)$ (see Lemma 5). Since the domain of the function is of size $(\lambda k)^{O(k^2)}$ and the tree decomposition is polynomial in the size of the input, the total number of sub-problems that we solve in the dynamic program is $(\lambda k)^{O(k^2)} n^{O(1)}$ (with $O(m)$ preprocessing time), thus proving Theorem 4.

In order to achieve the weaker goal of computing a function f_t for the tree node t that is f-correct and f-sound, we progressively define sub-problems and note that it suffices to achieve a weaker goal for all these sub-problems. Consequently, our goal reduces to computing Boolean functions that satisfy certain weaker properties. We encourage the reader to trace towards the base case of the dynamic program during the first read of the dynamic program.

One of the advantages of our dynamic program (in contrast to that of [23]) is that it is also applicable for alternative norm-based measures of k-partitions: here, the goal is to find a k-partition of the vertex set of the given edge-weighted graph so as to minimize $(\sum_{i=1}^{k} w(\delta(V_i))^p)^{1/p}$—we call this as MIN ℓ_p-NORM k-CUT. We note that MINMAX k-CUT is exactly MIN ℓ_∞-NORM k-CUT while MINSUM k-CUT is exactly MIN ℓ_1-NORM k-CUT. Our dynamic program can also be used to obtain the counterpart of Theorem 3 for MIN ℓ_p-NORM k-CUT in connected graphs for every $p \geq 1$. For disconnected graphs, we can get a $(1+\epsilon)$-approximation in time $(\lambda k/\epsilon)^{O(k^2)} n^{O(1)}$ by using the same ideas as in the Proof of Theorem 3 from Theorem 4 but by replacing every edge with $\lceil k/\epsilon \rceil$ copies. These results in conjunction with the reduction to unweighted instances (which can be shown to hold for MIN ℓ_p-NORM k-CUT) also leads to a parameterized approximation scheme for MIN ℓ_p-NORM k-CUT for every $p \geq 1$ in connected graphs.

Organization. We set up the tools to prove Theorem 4 in Sect. 2. We prove Theorem 3 in Sect. 3. We show a reduction from weighted instances to unweighted instances with logarithmic optimum value in Sect. 4. We use Theorem 4 and the reduction to unweighted instances with logarithmic optimum value to prove Theorem 2 in Sect. 5.

We prove the hardness results mentioned in Theorem 1 in Sect. 6. We conclude with a few open questions in Sect. 7.

2 Tools for the fixed-parameter algorithm

In this section, we set up the background for the fixed-parameter algorithm of Theorem 4. All graphs in this section and Sect. 3 could have parallel edges. Let $G = (V, E)$ be a graph. Throughout this work, we consider a partition to be an ordered tuple of non-empty subsets. An ordered tuple of subsets $(S_1, \ldots, S_k)$, where $S_i \subseteq V$ for all $i \in [k]$, is a k-subpartition of V if $S_1 \cup \ldots \cup S_k = V$ and $S_i \cap S_j = \emptyset$ for every pair of distinct $i, j \in [k]$. We emphasize the distinction between partitions and k-subpartitions—in a partition, all parts are required to be non-empty but the number of parts can be fewer than k while a k-subpartition allows for empty parts but the number of parts is exactly k.

For a subgraph $H \subseteq G$, a subset $X \subseteq V$, and a partition/k-subpartition $\mathcal{P}$ of X, we use $\delta_H(\mathcal{P})$ to denote the set of edges in $E(H)$ whose end-vertices are in different parts of $\mathcal{P}$. For a subgraph H of G and a subset $S \subseteq V(H)$, we use $\delta_H(S)$ to denote the set of edges in H with exactly one end-vertex in S. We will denote the set of (exclusive) neighbors of a subset S of vertices in the graph G by $N_G(S)$. We need the notion of a tree decomposition.

Definition 1 Let $G = (V, E)$ be a graph. A pair (τ, χ), where τ is a tree and $\chi : V(\tau) \to 2^V$ is a mapping of the nodes of the tree to subsets of vertices of the graph, is a *tree decomposition* of G if the following conditions hold:

(i) $\cup_{t \in V(\tau)} \chi(t) = V$,

(ii) for every edge $e = uv \in E$, there exists some $t \in V(\tau)$ such that $u, v \in \chi(t)$, and

(iii) for every $v \in V$, the set of nodes $\{t \in V(\tau) : v \in \chi(t)\}$ induces a connected subtree of τ.

For each $t \in V(\tau)$, we call $\chi(t)$ to be a *bag* of the tree decomposition.

We now describe certain notations that will be helpful while working with the tree decomposition. Let (τ, χ) be the tree decomposition of the graph $G = (V, E)$. We root τ at an arbitrary node $\Gamma \in V(\tau)$. For a tree node $t \in V(\tau) \backslash \{\Gamma\}$, there is a unique edge between t and its parent. Removing this edge disconnects τ into two subtrees τ_1 and τ_2, and we say that the set $A_t := \chi(\tau_1) \cap \chi(\tau_2)$ is the *adhesion associated with* t. For the root node Γ, we define $A_\Gamma := \emptyset$. For a tree node $t \in V(\tau)$, we denote the subgraph induced by all vertices in bags descending from t as G_t (here, the node t is considered to be a descendant of itself), i.e.,

$$G_t := G\left[\bigcup_{t' \text{ is a descendant of } t} \chi(t')\right].$$

We need the notions of compactness and edge-unbreakability.

Definition 2 A tree decomposition (τ, χ) of a graph G is *compact* if for every tree node $t \in V(\tau)$, the set of vertices $V(G_t)\backslash A_t$ induces a connected subgraph in G and $N_G(V(G_t)\backslash A_t) = A_t$.

Definition 3 Let $G = (V, E)$ be a graph and let $S \subseteq V$. The subset S is (a, b)-*edge-unbreakable* if for every nonempty proper subset S' of V satisfying $|E[S', V\backslash S']| \leq b$, we have that either $|S \cap S'| \leq a$ or $|S\backslash S'| \leq a$.

Informally, a subset S is (a, b)-edge-unbreakable if every non-trivial 2-partition of V either has large cut value or one side of the partition has small intersection with S. With these definitions, we have the following result from [23].

Lemma 1 [23,Theorem 4.1] *There exists an algorithm that takes a n-vertex m-edge graph $G = (V, E)$, an integer $k \geq 2$, and an integer λ as input, runs in time $poly(n, \lambda) + O(m)$, and returns a compact tree decomposition (τ, χ) of G such that*

(i) *each adhesion has size at most λk, and*
(ii) *for every tree node $t \in V(\tau)$, the bag $\chi(t)$ is $((\lambda k + 1)^5, \lambda k)$-edge-unbreakable.*

We observe that since Lemma 1 runs in polynomial time, the size of τ is necessarily polynomial in the input size. Next, we need the notion of α-respecting partitions.

Definition 4 Let $G = (V, E)$ be a graph and G' be a subgraph of G. A partition $\mathcal{P}$ of V α-*respects* G' if $|\delta_{G'}(\mathcal{P})| \leq \alpha$.

The following lemma will help us find a family of spanning trees of a given graph such that there exists an optimum k-partition that $(2k^2)$-respects some spanning tree in the family. The lemma follows from Lemma 4.3 in [6].

Lemma 2 [6] *Let G be a graph with optimum* MINSUM k-CUT *value OPT_{ms}. There exists a polynomial time algorithm that takes the graph G as input and returns a polynomial-sized family of spanning trees of G such that for each k-partition Π with $|\delta_G(\Pi)| \leq \alpha OPT_{ms}$, there exists a spanning tree T in the family with the property that Π $(2\alpha k)$-respects T.*

We recall the observation that an optimum solution to MINMAX k-CUT is a k-approximate optimum solution to MINSUM k-CUT. Hence, we have the following corollary from Lemma 2.

Corollary 1 *There exists a polynomial time algorithm that takes a graph G as input and returns a polynomial-sized family of spanning trees of G such that there exists an optimum min-max k-partition Π that $(2k^2)$-respects some spanning tree T in the family.*

We will frequently work with refinements and coarsenings of partitions and also restrictions of partitions to subsets.

Definition 5 Let $G = (V, E)$ be a graph, $S \subseteq V$ be a subset of vertices, and $k \geq 2$ be an integer.

1. Let $\mathcal{Q}$ be a partition/k-subpartition of S. A partition/k-subpartition $\mathcal{P}$ of S *coarsens* $\mathcal{Q}$ if each part of $\mathcal{P}$ is a union of parts of $\mathcal{Q}$.
2. Let $\mathcal{P}$ be a partition/k-subpartition of S. A partition/k-subpartition $\mathcal{Q}$ of S *refines* $\mathcal{P}$ if each part of $\mathcal{P}$ is a union of parts of $\mathcal{Q}$.
3. Let $\mathcal{P}$ be a partition/k-subpartition of V. A partition/k-subpartition $\mathcal{P}'$ of S is a *restriction of* $\mathcal{P}$ *to* S if for every $u, v \in S$, u and v are in the same part of $\mathcal{P}'$ if and only if they are in the same part of $\mathcal{P}$.

We note that we consider partitions/k-subpartitions as ordered tuples, and a *restriction of a partition/k-subpartition $\mathcal{P}$ to a subset S* is a reordering of the tuple obtained by taking the intersection of each part in $\mathcal{P}$ with S. Consequently, a partition/k-subpartition $\mathcal{P}$ can have multiple restrictions to a subset S. The following definition allows us to handle partitions of subsets that are crossed by a spanning tree at most $2k^2$ times.

Definition 6 Let $G = (V, E)$ be a graph, T be a spanning tree of G, and $X \subseteq V$. A partition $\mathcal{P}$ of X is T*-feasible* if there exists a partition $\mathcal{P}'$ of V such that

(i) $\mathcal{P}$ is a restriction of $\mathcal{P}'$ to X and
(ii) $\mathcal{P}'$ $(2k^2)$-respects T.

Moreover, a k-subpartition $\mathcal{P}'$ of X is T-feasible if the partition obtained from $\mathcal{P}'$ by discarding the empty parts of $\mathcal{P}'$ is T-feasible.

The next definition and the subsequent lemmas will show a convenient way to work with T-respecting partitions of a subset X of vertices, where T is a spanning tree.

Definition 7 Let $G = (V, E)$ be a graph, T be a spanning tree of G, and $X \subseteq V$. The graph $\mathrm{proj}(T, X)$ is the tree obtained from T by

1. repeatedly removing leaves of T that are not in X until there is none, and
2. for every path in T all of whose internal vertices are of degree 2 and are in $V \setminus X$, contract this path, i.e. replace each such path with a single edge, until there is none.

Observation 1 *Every vertex of $\mathrm{proj}(T, X)$ that is not in X has degree at least 3 in $\mathrm{proj}(T, X)$. Consequently, the number of vertices in $\mathrm{proj}(T, X)$ is $O(|X|)$.*

The next lemma gives a convenient way to work with T-feasible partitions of subsets of vertices. It is adapted and simplified from [23]. We give a proof for the sake of completeness. Throughout this work, we use tilde to indicate partitions/subpartitions of supersets of current sets of interest.

Lemma 3 *Let $G = (V, E)$ be a graph and T be a spanning tree of G.*

1. *For a subset $X \subseteq V(G)$ and a partition $\mathcal{P}$ of G, if $\mathcal{P}_X$ is a restriction of $\mathcal{P}$ to X, then there exists a partition $\tilde{\mathcal{P}}$ of the vertices of $\mathrm{proj}(T, X)$ such that $|\delta_{\mathrm{proj}(T,X)}(\tilde{\mathcal{P}})| \leq |\delta_T(\mathcal{P})|$ and $\mathcal{P}_X$ is a restriction of $\tilde{\mathcal{P}}$ to X.*
2. *For a subset $X \subseteq V$ and a partition $\tilde{\mathcal{P}}$ of the vertices of $\mathrm{proj}(T, X)$, if $\tilde{\mathcal{P}}_X$ is a restriction of $\tilde{\mathcal{P}}$ to X, then there exists a partition $\mathcal{P}$ of $V(G)$ such that $|\delta_T(\mathcal{P})| \leq |\delta_{\mathrm{proj}(T,X)}(\tilde{\mathcal{P}})|$ and $\tilde{\mathcal{P}}_X$ is a restriction of $\mathcal{P}$ to X.*

Proof We will start by proving the first statement. Let $X \subseteq V(G)$ and $\mathcal{P}$ be fixed as in the first statement. We will construct $\tilde{\mathcal{P}}$ by constructing $\delta_{\text{proj}(T,X)}(\tilde{\mathcal{P}})$ as follows. For each edge $e \in \delta_T(\mathcal{P})$, if e remains in $\text{proj}(T, X)$, then include e into $\delta_{\text{proj}(T,X)}(\tilde{\mathcal{P}})$; if e is removed as part of a path that is replaced with an edge $e' \in E(\text{proj}(T, X))$, then include e' into $\delta_{\text{proj}(T,X)}(\tilde{\mathcal{P}})$. By doing this, we can guarantee that $|\delta_{\text{proj}(T,X)}(\tilde{\mathcal{P}})| \leq |\delta_T(\mathcal{P})|$.

To see that $\delta(\tilde{\mathcal{P}})$ indeed yields a partition whose restriction to X is $\mathcal{P}_X$, we claim that the partition $\tilde{\mathcal{P}}'$ whose parts are the connected components of $\text{proj}(T, X)\backslash\delta_{\text{proj}(T,X)}(\tilde{\mathcal{P}})$, when restricted to X, refines $\mathcal{P}_X$. For every pair of vertices $u, v \in X$ that are in different parts of $\mathcal{P}_X$, we know u and v are also in different parts of $\mathcal{P}$. This means some edge e on the unique path in T between u and v is in $\delta_T(\mathcal{P})$. By construction of $\delta_{\text{proj}(T,X)}(\tilde{\mathcal{P}})$, either e is contained in $\delta_{\text{proj}(T,X)}(\tilde{\mathcal{P}})$, or e' which replaces a path containing e is contained in $\delta_{\text{proj}(T,X)}(\tilde{\mathcal{P}})$. In either case, the unique path in $\text{proj}(T, X)$ between u and v is disconnected. This proves our claim. To construct $\tilde{\mathcal{P}}$ from $\delta_{\text{proj}(T,X)}(\tilde{\mathcal{P}})$, we can group parts of $\tilde{\mathcal{P}}'$ together as necessary to comply with $\mathcal{P}_X$ because $\tilde{\mathcal{P}}'$ refines $\mathcal{P}_X$. This completes the proof of the first statement.

The proof of the second statement is similar to the preceding proof. Let $X \subseteq V$ and $\tilde{\mathcal{P}}$ be fixed as in the second statement. We start by constructing the edge set $\delta_T(\mathcal{P})$. For each edge $e \in \delta_{\text{proj}(T,X)}(\tilde{\mathcal{P}})$, if e is originally an edge of T, then include e into $\delta_T(\mathcal{P})$; if e is introduced to replace a path in T, then fix an arbitrary edge e' in this path and include e' into $\delta_T(\mathcal{P})$. By doing this, clearly we can guarantee $|\delta_T(\mathcal{P})| \leq |\delta_{\text{proj}(T,X)}(\tilde{\mathcal{P}})|$.

By the same argument as in proof of the first statement, we can see that the connected components of $T\backslash\delta_T(\mathcal{P})$ yields a partition that, when restricted to X, refines $\tilde{\mathcal{P}}_X$. Combining parts as necessary, we obtain a desired partition $\mathcal{P}$. This completes the proof of the second statement. $\square$

3 Fixed-parameter algorithm parameterized by *k* and solution size

In this section we prove Theorem 4. Let $(G = (V, E), k)$ be the input instance of Minmax k-cut with n vertices. The input graph G could possibly have parallel edges. The following will be the main theorem of this section.

Theorem 4 *There exists an algorithm that takes as input an n-vertex m-edge unweighted connected graph $G = (V, E)$ and an integer $k \geq 2$, along with an integer λ, and runs in time $(k\lambda)^{O(k^2)}n^{O(1)} + O(m)$ to determine if there exists a k-partition $(V_1, \ldots, V_k)$ of V such that $cost_G(V_1, \ldots, V_k) \leq \lambda$ and if so, returns an optimum partition for* Minmax *k-cut on G.*

We now complete the Proof of Theorem 3 using Theorem 4.

Theorem 3 *There exists an algorithm that takes as input an unweighted instance $G = (V, E)$ of* Minmax *k-cut, namely an n-vertex m-edge graph $G = (V, E)$ and an integer $k \geq 2$, along with an integer λ and runs in time $(k\lambda)^{O(k^2)}n^{O(1)} + O(km)$ to determine if there exists a k-partition $(V_1, \ldots, V_k)$ of V such that $cost_G(V_1, \ldots, V_k) \leq \lambda$ and if so, returns an optimum partition for* Minmax *k-cut on G.*

Proof If G is connected, then this follows from Theorem 4. For the rest of the proof, we assume that G is disconnected. We note that in a disconnected graph, it is not guaranteed that each part in an optimum solution of MINMAX k-CUT is contained within a single connected component (see one such example in Fig. 2). So, we need to address disconnected instances with a little more care.

Let $C_1, C_2, \ldots, C_t$ be the connected components in G. We may assume that $t < k$, since $\text{OPT}(G, k) = 0$ otherwise. We construct a connected graph $H = (V, E')$ with the same vertex set V as follows:

1. For each edge $e \in E$, we add t parallel edges between the end vertices of e and
2. we add $t - 1$ arbitrary edges between the components $C_1, C_2, \ldots, C_t$ so that H becomes connected.

We run the algorithm from Theorem 4 on the instance $(H, k, k(\lambda + 1))$ and return its output.

We now analyze the run-time of the algorithm. The graph H can be constructed in time $\mathsf{O}(km)$. The run-time of the algorithm from Theorem 4 on the instance $(H, k, k(\lambda+1))$ is $(k^2(\lambda+1))^{\mathsf{O}(k^2)} n^{\mathsf{O}(1)} + \mathsf{O}(km) = (k\lambda)^{\mathsf{O}(k^2)} n^{\mathsf{O}(1)} + \mathsf{O}(km)$. Hence, the overall run-time is $(k\lambda)^{\mathsf{O}(k^2)} n^{\mathsf{O}(1)} + \mathsf{O}(km)$.

Next, we prove the correctness of the algorithm. Suppose that there exists a k-partition of V such that $\text{cost}_G(V_1, \ldots, V_k) \le \lambda$. We need to show that the algorithm from Theorem 4 on the instance $(H, k, k(\lambda+1))$ indeed returns YES and the partition $(V_1, \ldots, V_k)$ returned by the algorithm is in fact an optimum k-partition in G.

Let $(P_1, \ldots, P_k)$ be a k-partition of V that corresponds to an optimum MINMAX k-CUT in G. Then, we have that for each $i \in [k]$,

$$|\delta_H(P_i)| \le t|\delta_G(P_i)| + t - 1 \le t\text{OPT}(G, k) + t - 1.$$

Consequently,

$$\text{OPT}(H, k) \le t\text{OPT}(G, k) + t - 1. \tag{1}$$

In particular, $\text{OPT}(H, k) \le t\text{OPT}(G, k) + t - 1) \le t(\text{OPT}(G, k) + 1) < k(\lambda + 1)$ and hence, the algorithm from Theorem 4 on the instance $(H, k, k(\lambda + 1))$ indeed returns YES.

Next, let $(Q_1, \ldots, Q_k)$ be the optimum k-partition in H that is returned by the algorithm from Theorem 4 on the instance $(H, k, k(\lambda + 1))$. Then,

$$\text{OPT}(H, k) \ge |\delta_H(Q_i)| \ge t|\delta_G(Q_i)|.$$

This implies that for every $i \in [k]$,

$$|\delta_G(Q_i)| \le \frac{1}{t}\text{OPT}(H, k) \le \frac{1}{t}(t\text{OPT}(G, k) + t - 1) = \text{OPT}(G, k) + \frac{t-1}{t}.$$

The second inequality above is by using inequality (1). As a consequence, we have that

$$\max_{i\in[k]} |\delta_G(Q_i)| \le \mathrm{OPT}(G,k) + \frac{t-1}{t}.$$

Since $\max_{i\in[k]} |\delta_G(Q_i)|$ and $\mathrm{OPT}(G,k)$ are both integral, we conclude that $\max_{i\in[k]} |\delta_G(Q_i)| = \mathrm{OPT}(G,k)$ and hence, the k-partition $(Q_1,\ldots,Q_k)$ is an optimum MINMAX k-CUT in G. □

The rest of the section will be devoted to proving Theorem 4. For this purpose, we assume that G is connected. Let $\mathrm{OPT} = \mathrm{OPT}(G,k)$ (i.e., OPT is the optimum objective value of MINMAX k-CUT on input G) and let λ be the input such that $\lambda \ge \mathrm{OPT}$. We will design a dynamic programming algorithm that runs in time $(\lambda k)^{O(k^2)} n^{O(1)} + O(m)$ to compute OPT.

Given the input, we first use Lemma 1 to obtain a tree decomposition (τ,χ) of G satisfying the conditions of the lemma. Since the algorithm in the lemma runs in polynomial time, the size of the tree decomposition (τ,χ) is polynomial in the input size. Next, we use Corollary 1 to obtain a polynomial-sized family of spanning trees such that there exists an optimum min-max k-partition Ω of V that $(2k^2)$-respects a spanning tree T in the family, and moreover T is a subgraph of G. The rest of our algorithm would iterate over each spanning tree in the family. In the rest of this section, we fix a spanning tree T such that there exists an optimum min-max k-partition $\Omega = (\Omega_1,\ldots,\Omega_k)$ of V that $(2k^2)$-respects T. We fix the tree decomposition (τ,χ), the spanning tree T, and the optimum solution Ω with these choices in the rest of this section. We note that $\Omega_i \neq \emptyset$ for all $i \in [k]$ and $\max_{i\in[k]} |\delta_G(\Omega_i)| = \mathrm{OPT}$. We emphasize that the choice of Ω is fixed only for the purposes of the correctness of the algorithm and is not known to the algorithm explicitly. We will preprocess the graph G (that possibly has parallel edges) in $O(m+n^2)$ time to compute the number of edges between every pair of vertices; this information will suffice to compute the graph cut function value for a given set in time $n^{O(1)}$. The rest of our algorithm can be implemented with access to the graph cut function oracle and will run in time $(\lambda k)^{O(k^2)} n^{O(1)}$.

Our algorithm is based on dynamic program (DP). We will describe the subproblems of the DP in Sect. 3.1. We will need the notion of a nice decomposition of the bags corresponding to the tree decomposition. We describe this notion in Sect. 3.2 and give an algorithm to generate them in Sect. 3.4. We will show the recursion to solve the dynamic program in Sect. 3.3. We encourage the reader to trace towards the base case of the dynamic program on first read.

3.1 Subproblems of the DP

In this section, we state the subproblems in our dynamic program (DP), bound the number of subproblems in the DP, and prove Theorem 4. For a tree node $t \in V(\tau)$, let $\mathcal{F}^{A_t}$ be the collection of partitions of the adhesion A_t that are (i) T-feasible and (ii) have at most k parts. We emphasize that elements of $\mathcal{F}_{A_t}$ are of the form $\mathcal{P}_{A_t} =$

$(\tilde{P}_1, \ldots, \tilde{P}_{k'})$ for some $k' \in \{0, 1, \ldots, k\}$, where $\tilde{P}_i \neq \emptyset$ for all $i \in [k']$. The following lemma bounds the size of $\mathcal{F}^{A_t}$, which in turn, will be helpful in bounding the number of subproblems to be solved in our dynamic program.

Lemma 4 *For every tree node $t \in V(\tau)$, we have $|\mathcal{F}^{A_t}| = (\lambda k)^{\mathsf{O}(k^2)}$. Moreover, the collection $\mathcal{F}^{A_t}$ can be enumerated in $(\lambda k)^{\mathsf{O}(k^2)}$ time.*

Proof First we claim that a partition $\mathcal{P}_{A_t}$ of A_t is T-feasible if and only if it is $\mathrm{proj}(T, A_t)$-feasible.

Assume a partition $\mathcal{P}_{A_t}$ of A_t is T-feasible, realized by a partition $\mathcal{P}$ of V. It follows that $|\delta_T(\mathcal{P})| \leq 2k^2$. By Lemma 3, there exists a partition $\tilde{\mathcal{P}}$ of $\mathrm{proj}(T, A_t)$ such that $|\delta_{\mathrm{proj}(T,A_t)}(\tilde{\mathcal{P}})| \leq 2k^2$ and $\mathcal{P}_{A_t}$ is a restriction of $\tilde{\mathcal{P}}$ to A_t. This is equivalent to saying $\mathcal{P}_{A_t}$ is $\mathrm{proj}(T, A_t)$-feasible.

The other direction is similar. If a partition $\mathcal{P}_{A_t}$ of A_t is $\mathrm{proj}(T, A_t)$-feasible, realized by a partition $\tilde{\mathcal{P}}$ of $V(\mathrm{proj}(T, A_t))$, it follows that $|\delta_{\mathrm{proj}(T,A_t)}(\tilde{\mathcal{P}})| \leq 2k^2$. By Lemma 3 there exists a partition $\mathcal{P}$ of G such that $|\delta_T(\mathcal{P})| \leq 2k^2$ and $\mathcal{P}_{A_t}$ is a restriction of $\mathcal{P}$ to A_t. Hence $\mathcal{P}_{A_t}$ is T-feasible.

It remains to bound the number of $\mathrm{proj}(T, A_t)$-feasible partitions of A_t. By Observation 1, the size of $\mathrm{proj}(T, A_t)$ is $\mathsf{O}(\lambda k)$. We notice that partitions with at most k parts that $2k^2$-respect $\mathrm{proj}(T, A_t)$ can be enumerated by removing up to $2k^2$ edges of $\mathrm{proj}(T, A_t)$, and putting the resulting connected components (there are at most $2k^2+1$ of them) into k bins. Therefore, combining the previous observation, we conclude that

$$|\mathcal{F}^{A_t}| = \mathsf{O}(\lambda k)^{2k^2} \cdot k^{2k^2+1} = (\lambda k)^{\mathsf{O}(k^2)}.$$

Moreover, the time required to compute $\mathcal{F}^{A_t}$ (by enumerating all eligible partitions as above) is also $(\lambda k)^{\mathsf{O}(k^2)}$. □

The following definition will be useful in identifying the subproblems of the DP.

Definition 8 Let $t \in V(\tau)$ be a tree node, and $f_t : \mathcal{F}^{A_t} \times \{0, 1, \ldots, \lambda\}^k \times \{0, 1, \ldots, 2k^2\} \to \{0, 1\}$ be a Boolean function.

1. **(Correctness)** The function f_t is *f-correct* if we have $f_t(\mathcal{P}_{A_t}, \bar{x}, d) = 1$ for all $\mathcal{P}_{A_t} = (\tilde{P}_1, \ldots, \tilde{P}_{k'}) \in \mathcal{F}^{A_t}$, $\bar{x} = (x_1, \ldots, x_k) \in \{0, 1, \ldots, \lambda\}^k$, and $d \in \{0, 1, \ldots, 2k^2\}$ for which there exists a k-subpartition $\mathcal{P} = (P'_1, \ldots, P'_k)$ of $V(G_t)$ satisfying the following conditions:

   (i) $P'_i \cap A_t = \tilde{P}_i$ for all $i \in [k']$,
   (ii) $|\delta_{G_t}(P'_i)| = x_i$ for all $i \in [k]$,
   (iii) $|\delta_T(\mathcal{P})| \leq d$, and
   (iv) $\mathcal{P}$ is a restriction of Ω to $V(G_t)$.

   A k-subpartition of $V(G_t)$ satisfying the above four conditions is said to *witness* f-correctness of $f_t(\mathcal{P}_{A_t}, \bar{x}, d)$.
2. **(Soundness)** The function f_t is *f-correct* if for all $\mathcal{P}_{A_t} = (\tilde{P}_1, \ldots, \tilde{P}_{k'}) \in \mathcal{F}^{A_t}$, $\bar{x} = (x_1, \ldots, x_k) \in \{0, 1, \ldots, \lambda\}^k$ and $d \in \{0, 1, \ldots, 2k^2\}$, we have $f_t(\mathcal{P}_{A_t}, \bar{x}, d) = 1$ only if there exists a k-subpartition $\mathcal{P} = (P'_1, \ldots, P'_k)$ of $V(G_t)$

satisfying conditions (i), (ii) and (iii) above. A k-subpartition of $V(G_t)$ satisfying (i), (ii) and (iii) is said to *witness* f-soundness of $f_t(\mathcal{P}_{A_t}, \bar{x}, d)$.

We emphasize the distinction between correctness and soundness: correctness relies on all four conditions while soundness relies only on three conditions. Correctness guarantees that the function value of f_t is set to be 1 on inputs obtained by restricting Ω, and soundness guarantees that a partition of V (not necessarily Ω) that satisfies (i), (ii) and (iii) indeed exists when f_t is set to 1. We discuss the need for distinct correctness and soundness definitions after Lemma 6.

The next proposition shows that an f-correct and f-sound function for the root node of the tree decomposition can be used to recover the optimum value.

Proposition 1 *If we have a function $f_\Gamma : \mathcal{F}^{A_\Gamma} \times \{0, 1, \dots, \lambda\}^k \times \{0, 1, \dots, 2k^2\} \to \{0, 1\}$ that is both f-correct and f-sound, where Γ is the root of the tree decomposition τ, then*

$$OPT = \min\left\{\max_{i\in[k]}\{x_i\} : f_\Gamma(\mathcal{P}_\emptyset, \bar{x}, 2k^2) = 1, \bar{x} = (x_1, \dots, x_k) \in [\lambda]^k\right\},$$

where $\mathcal{P}_\emptyset$ is the 0-tuple that denotes the trivial partition of $A_\Gamma = \emptyset$.

Proof The optimum partition Ω is a k-subpartition of $V = V(G_\Gamma)$ that witnesses f-correctness of $f_\Gamma(\mathcal{P}_\emptyset, (|\delta_G(\Omega_1)|, \dots, |\delta_G(\Omega_k)|), 2k^2)$. Hence, $f_\Gamma(\mathcal{P}_\emptyset, (|\delta_G(\Omega_1)|, \dots, |\delta_G(\Omega_k)|), 2k^2) = 1$, where $|\delta_G(\Omega_i)| \in [\lambda]$ for every $i \in [k]$. Consequently,

$$\min\left\{\max_{i\in[k]}\{x_i\} : f_\Gamma(\mathcal{P}_\emptyset, \bar{x}, 2k^2) = 1, \bar{x} \in [\lambda]^k\right\} \le \max_{i\in[k]}\{|\delta_G(\Omega_i)|\} = \mathrm{OPT}.$$

We now show the reverse inequality. Suppose that $f_\Gamma(\mathcal{P}_\emptyset, \bar{x}, 2k^2) = 1$ for some $\bar{x} \in [\lambda]^k$. Then there exists a k-subpartition $\mathcal{P}'$ of V witnessing f-soundness of $f_\Gamma(\mathcal{P}_\emptyset, \bar{x}, 2k^2)$. Since $\bar{x} \in [\lambda]^k$, we know that $x_i \ge 1$ for every $i \in [k]$. Since G is connected, this implies that each part of $\mathcal{P}'$ is non-empty. Therefore, $\mathcal{P}'$ is also a k-partition of V and is hence, feasible for MINMAX k-CUT. This implies that

$$\max_{i\in[k]}\{x_i\} \ge \mathrm{OPT}. \qquad \square$$

By Proposition 1, it suffices to compute an f-correct and f-sound function f_Γ, where Γ is the root of the tree decomposition τ. We will compute this in a bottom-up fashion on the tree decomposition using the following lemma.

Lemma 5 *There exists an algorithm that takes as input (τ, χ), a tree node $t \in V(\tau)$, Boolean functions $f_{t'} : \mathcal{F}^{A_{t'}} \times \{0, 1, \dots, \lambda\}^k \times \{0, 1, \dots, 2k^2\} \to \{0, 1\}$ for every child t' of t in τ that are f-correct and f-sound, and runs in time $(\lambda k)^{O(k^2)} n^{O(1)}$ to return a function $f_t : \mathcal{F}^{A_t} \times \{0, 1, \dots, \lambda\}^k \times \{0, 1, \dots, 2k^2\} \to \{0, 1\}$ that is f-correct and f-sound.*

We now complete the proof of Theorem 4 using Lemma 5 and Proposition 1.

***Proof of Theorem* 4** In order to compute a function $f_\Gamma : \mathcal{F}^{A_\Gamma} \times \{0, 1, \dots, \lambda\}^k \times \{0, 1, \dots, 2k^2\} \to \{0, 1\}$ that is both f-correct and f-sound, we can apply Lemma 5 on each tree node $t \in V(\tau)$ in a bottom up fashion starting from the leaf nodes of the tree decomposition. Therefore, using Lemmas 4 and 5, the total run time to compute f_Γ is

$$(\lambda k)^{\mathrm{O}(k^2)} n^{\mathrm{O}(1)} \cdot |V(\tau)| = (\lambda k)^{\mathrm{O}(k^2)} n^{\mathrm{O}(1)} \cdot \mathrm{poly}(n, \lambda, k) = (\lambda k)^{\mathrm{O}(k^2)} n^{\mathrm{O}(1)}.$$

Using Proposition 1, we can compute OPT from the function f_Γ. Consequently, the total time to compute OPT is $(\lambda k)^{\mathrm{O}(k^2)} n^{\mathrm{O}(1)}$. □

We will prove Lemma 5 in the following subsections. We fix the tree node $t \in V(\tau)$ for the rest of the subsections.

3.2 Nice decomposition

We need the notion of a nice decomposition that we define below. Our definition differs from the notion of the nice decomposition defined by [23] in property (ii) below (we use $4k^2 + 1$ while [23] use $2k - 1$).

Definition 9 A *nice decomposition* of $\chi(t)$ is a triple $(\mathcal{P}_{\chi(t)}, \mathcal{Q}_{\chi(t)}, O)$ where $\mathcal{P}_{\chi(t)}$ and $\mathcal{Q}_{\chi(t)}$ are partitions of $\chi(t)$, $\mathcal{Q}_{\chi(t)}$ refines $\mathcal{P}_{\chi(t)}$, and O is either a part of $\mathcal{P}_{\chi(t)}$ or $\emptyset$. Additionally, the following conditions need to be met:

(i) If $O \neq \emptyset$, then O is a part of $\mathcal{Q}_{\chi(t)}$.
If $O = \emptyset$, then $\mathcal{P}_{\chi(t)}$ has only one part.
(ii) For every part P of $\mathcal{P}_{\chi(t)}$, P contains at most $4k^2 + 1$ parts of $\mathcal{Q}_{\chi(t)}$.
(iii) For every pair of distinct parts P, P' of $\mathcal{P}_{\chi(t)}$ other than O, there are no edges between P and P'.
(iv) If $t' \in V(\tau)$ is a child of t or t itself, then $A_{t'}$ intersects with at most one part of $\mathcal{P}_{\chi(t)}$ other than O.

In order to compute an f-correct and f-sound function $f_t : \mathcal{F}^{A_t} \times \{0, 1, \dots, \lambda\}^k \times \{0, 1, \dots, 2k^2\} \to \{0, 1\}$, we will compute a family $\mathcal{D}$ of nice decompositions of $\chi(t)$ such that if there exists a k-subpartition Π of $V(G_t)$ that realizes f-correctness of $f_t(\mathcal{P}_{A_t}, \bar{x}, d)$ for some $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, $\bar{x} \in \{0, 1, \dots, \lambda\}^k$ and $d \in \{0, 1, \dots, 2k^2\}$, then there exists a nice decomposition $(\mathcal{P}_{\chi(t)}, \mathcal{Q}_{\chi(t)}, O)$ in $\mathcal{D}$ such that $\mathcal{Q}_{\chi(t)}$ refines a restriction of Π to $\chi(t)$. A formal statement is given in Lemma 6.

Lemma 6 *There exists an algorithm that takes as input the spanning tree T, the tree decomposition (τ, χ), a tree node $t \in V(\tau)$, and runs in time $(\lambda k)^{\mathrm{O}(k^2)} n^{\mathrm{O}(1)}$ to return a family $\mathcal{D}$ of nice decompositions of $\chi(t)$ with $|\mathcal{D}| = (\lambda k)^{\mathrm{O}(k^2)} n^{\mathrm{O}(1)}$. Additionally, if a k-subpartition Π of $V(G_t)$ realizes f-correctness of $f_t(\mathcal{P}_{A_t}, \bar{x}, d)$ for some $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, $\bar{x} \in \{0, 1, \dots, \lambda\}^k$ and $d \in \{0, 1, \dots, 2k^2\}$, then $\mathcal{D}$ contains a nice decomposition $(\mathcal{P}_{\chi(t)}, \mathcal{Q}_{\chi(t)}, O)$ where $\mathcal{Q}_{\chi(t)}$ refines a restriction of Π to $\chi(t)$.*

We defer the Proof of Lemma 6 to Sect. 3.4.

We now discuss the need for distinct definitions for correctness and soundness. If a k-subpartition Π of $V(G_t)$ realizes f-correctness of $f_t(\mathcal{P}_{A_t}, \bar{x}, d)$ for some $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, $\bar{x} \in \{0, 1, \dots, \lambda\}^k$ and $d \in \{0, 1, \dots, 2k^2\}$, then Π is a restriction of the optimal partition Ω to $V(G_t)$. The family $\mathcal{D}$ of nice decompositions then serves to provide a partition $\mathcal{Q}_{\chi(t)}$ of $\chi(t)$ that refines a restriction of the optimal partition Ω to $\chi(t)$, which will later be used to identify an optimal partition. We note that if a k-subpartition Π of $V(G_t)$ only witnesses f-soundness of $f_t(\mathcal{P}_{A_t}, \bar{x}, d)$ for some $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, $\bar{x} \in \{0, 1, \dots, \lambda\}^k$ and $d \in \{0, 1, \dots, 2k^2\}$, then the family $\mathcal{D}$ is not guaranteed to provide a refinement of a restriction of the k-subpartition Π to $\chi(t)$. This motivates the two distinct definitions for correctness and soundness.

3.3 Computing an *f*-correct and *f*-sound function f_t

In this section, we will prove Lemma 5. For a fixed tree node $t \in V(\tau)$, we will describe an algorithm to assign values to $f_t(\mathcal{P}_{A_t}, \bar{x}, d)$ for all $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, $\bar{x} \in \{0, 1, \dots, \lambda\}^k$, and $d \in \{0, 1, \dots, 2k^2\}$ based on the value of $f_{t'}(\mathcal{P}_{A_{t'}}, \bar{\alpha}, \beta)$ for all children t' of t, and all $\mathcal{P}_{A_{t'}} \in \mathcal{F}^{A_{t'}}$, $\bar{\alpha} \in \{0, 1, \dots, \lambda\}^k$, and $\beta \in \{0, 1, \dots, 2k^2\}$ so that the resulting function f_t is f-correct and f-sound.

For the fixed $t \in V(\tau)$, we use Lemma 6 to obtain a family $\mathcal{D}$ of nice decompositions of $\chi(t)$. Our plan to compute the function f_t involves working with each nice decomposition $D \in \mathcal{D}$. The following definition will be helpful to achieve our goal of computing the function f_t.

Definition 10 Let $D := (\mathcal{P}_{\chi(t)}, \mathcal{Q}_{\chi(t)}, O) \in \mathcal{D}$, and $r^D : \mathcal{F}^{A_t} \times [\lambda]^k \times \{0, 1, \dots, 2k^2\} \to \{0, 1\}$ be a Boolean function.

1. **(Correctness)** The function r^D is *r-correct* if we have $r^D(\mathcal{P}_{A_t}, \bar{x}, d) = 1$ for all $\mathcal{P}_{A_t} = (\tilde{P}_1, \dots, \tilde{P}_{k'}) \in \mathcal{F}^{A_t}$, $\bar{x} = (x_1, \dots, x_k) \in \{0, 1, \dots, \lambda\}^k$ and $d \in \{0, 1, \dots, 2k^2\}$ for which there exists a k-subpartition $\mathcal{P} = (P'_1, \dots, P'_k)$ of $V(G_t)$ satisfying the following conditions:

   (i) $P'_i \cap A_t = \tilde{P}_i$ for all $i \in [k']$,
   (ii) $|\delta_{G_t}(P'_i)| = x_i$ for all $i \in [k]$,
   (iii) $|\delta_T(\mathcal{P})| \le d$,
   (iv) $\mathcal{Q}_{\chi(t)}$ refines $\mathcal{P}$ restricted to $\chi(t)$, and
   (v) $\mathcal{P}$ is a restriction of Ω to $V(G_t)$.

   A k-subpartition of $V(G_t)$ satisfying the above five conditions is said to *witness* r-correctness of $r^D(\mathcal{P}_{A_t}, \bar{x}, d)$.
2. **(Soundness)** The function r^D is *r-correct* if for all $\mathcal{P}_{A_t} = (\tilde{P}_1, \dots, \tilde{P}_{k'}) \in \mathcal{F}^{A_t}$, $\bar{x} = (x_1, \dots, x_k) \in \{0, 1, \dots, \lambda\}^k$ and $d \in \{0, 1, \dots, 2k^2\}$, we have $r^D(\mathcal{P}_{A_t}, \bar{x}, d) = 1$ only if there exists a k-subpartition $\mathcal{P} = (P'_1, \dots, P'_k)$ of $V(G_t)$ satisfying conditions (i), (ii), (iii) and (iv) above. A k-subpartition of $V(G_t)$ satisfying (i), (ii), (iii) and (iv) is said to *witness* r-soundness of $r^D(\mathcal{P}_{A_t}, \bar{x}, d)$.

We note that the only difference between r-correctness/r-soundness and f-correctness/f-soundness is that r-correctness/r-soundness has the additional condition (iv). The next proposition shows that r-correct and r-sound functions can be used to recover an f-correct and f-correct function.

Proposition 2 *Suppose that we have functions $r^D : \mathcal{F}^{A_t} \times [\lambda]^k \times \{0, 1, \dots, 2k^2\} \to \{0, 1\}$ for every $D \in \mathcal{D}$ such that all of them are both r-correct and r-sound. Then, the function f_t obtained by setting*

$$f_t(\mathcal{P}_{A_t}, \bar{x}, d) := \max\{r^D(\mathcal{P}_{A_t}, \bar{x}, d) : \ D \in \mathcal{D}\}$$

for every $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, $\bar{x} \in \{0, 1, \dots, \lambda\}^k$ and $d \in \{0, 1, \dots, 2k^2\}$ is both f-correct and f-sound.

Proof We first show f-correctness. For $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, $\bar{x} \in \{0, 1, \dots, \lambda\}^k$ and $d \in \{0, 1, \dots, 2k^2\}$, suppose that there exists a k-subpartition Π of $V(G_t)$ witnessing f-correctness of $f_t(\mathcal{P}_{A_t}, \bar{x}, d)$. By Lemma 6, we know that $\mathcal{D}$ contains a nice decomposition $D^* = (\mathcal{P}^*_{\chi(t)}, \mathcal{Q}^*_{\chi(t)}, O^*)$ such that $\mathcal{Q}^*_{\chi(t)}$ refines Π restricted to $\chi(t)$. Then by r-correctness of r^{D^*}, we know that $r^{D^*}(\mathcal{P}_{A_t}, \bar{x}, d) = 1$. This implies $f_t(\mathcal{P}_{A_t}, \bar{x}, d) = 1$.

Next we show f-soundness. Suppose that $f_t(\mathcal{P}_{A_t}, \bar{x}, d) = 1$ for some $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, $\bar{x} \in \{0, 1, \dots, \lambda\}^k$ and $d \in \{0, 1, \dots, 2k^2\}$. Then, there exists $D \in \mathcal{D}$ such that $r^D(\mathcal{P}_{A_t}, \bar{x}, d) = 1$. By r-soundness of the function r^D, there exists a k-subpartition Π' of $V(G_t)$ witnessing r-soundness of $r^D(\mathcal{P}_{A_t}, \bar{x}, d)$. It follows by definition of r^D and f that Π' also witnesses f-soundness of $f_t(\mathcal{P}_{A_t}, \bar{x}, d)$. □

Our goal now is to compute an r-correct and r-sound function r^D for each $D \in \mathcal{D}$.

Lemma 7 *There exists an algorithm that takes as input (τ, χ), a tree node $t \in V(\tau)$, a nice decomposition $D = (\mathcal{P}_{\chi(t)}, \mathcal{Q}_{\chi(t)}, O) \in \mathcal{D}$, together with Boolean functions $f_{t'} : \mathcal{F}^{A_{t'}} \times \{0, 1, \dots, \lambda\}^k \times \{0, 1, \dots, 2k^2\} \to \{0, 1\}$ for every child t' of t that are f-correct and f-sound, and runs in time $(\lambda k)^{O(k^2)} n^{O(1)}$ to return a function $r^D : \mathcal{F}^{A_t} \times [\lambda]^k \times \{0, 1, \dots, 2k^2\} \to \{0, 1\}$ that is r-correct and r-sound.*

We now use Lemma 7 to complete the proof of Lemma 5.

Proof of Lemma 5 Using Lemma 6, we compute a family $\mathcal{D}$ of nice decompositions in time $(\lambda k)^{O(k^2)} n^{O(1)}$, where $|\mathcal{D}| = (\lambda k)^{O(k^2)} n^{O(1)}$. For each $D \in \mathcal{D}$, we use Lemma 7 to compute a function $r^D : \mathcal{F}^{A_t} \times [\lambda]^k \times \{0, 1, \dots, 2k^2\} \to \{0, 1\}$ that is both r-correct and r-sound in time $(\lambda k)^{O(k^2)} n^{O(1)}$. Finally, we use Proposition 2 to compute the desired function f_t that is both f-correct and f-sound. The total run-time is

$$(\lambda k)^{O(k^2)} n^{O(1)} + (\lambda k)^{O(k^2)} n^{O(1)} \cdot (\lambda k)^{O(k^2)} n^{O(1)} = (\lambda k)^{O(k^2)} n^{O(1)}. \qquad \square$$

The rest of the section is devoted to proving Lemma 7. We fix the inputs specified in Lemma 7 for the rest of this section. In particular, we additionally fix $D = (\mathcal{P}_{\chi(t)}, \mathcal{Q}_{\chi(t)}, O) \in \mathcal{D}$.

Notation. Let $\mathcal{P}_{\chi(t)} = (P_1, \dots, P_p, O)$. If $O = \emptyset$, we will abuse notation and use $\mathcal{P}_{\chi(t)}$ to refer to the partition of $\chi(t)$ containing only one part, namely $\mathcal{P}_{\chi(t)} = (P_1)$.

We note that $p \leq n$ since D is a nice decomposition. We define $P_{\leq \ell} := \cup_{i=1}^{\ell} P_i$, where $\ell \in \{0, 1, \ldots, p\}$. We will use $P_{\leq 0} := \emptyset$. We specially define $P_0 := O$ for indexing convenience. For every $\ell \in \{0, 1, \ldots, p\}$, we define $\mathcal{A}(\ell)$ to be the set of children of t whose adhesion is contained in $O \cup P_\ell$ and intersects P_ℓ, i.e.,

$$\mathcal{A}(\ell) := \{t' \in V(\tau) : t' \text{ is a child of } t,\, A_{t'} \subseteq O \cup P_\ell,\, A_{t'} \cap P_\ell \neq \emptyset\}.$$

Moreover, let $\mathcal{A}(\ell) := \{t_1, t_2, ..., t_{|\mathcal{A}(\ell)|}\}$. For each $\ell \in \{0, 1, \ldots, p\}$ and $a \in \{0, 1, \ldots, |\mathcal{A}(\ell)|\}$, let

$$G_{\leq}(\ell, a) := G\left[O \cup P_\ell \cup \bigcup_{1 \leq i \leq a} V(G_{t_i})\right],$$

$$G(\ell) := G\left[O \cup P_\ell \cup \bigcup_{t' \in \mathcal{A}(\ell)} V(G_{t'})\right], \text{ and}$$

$$G_{\leq}(\ell) := G\left[\bigcup_{0 \leq i \leq \ell} V(G(i))\right].$$

These subgraphs are illustrated in Figs. 3, 4, and 5. In order to compute an r-correct and r-sound function r^D, we will employ new sub-problems that we define below.

Definition 11 Let $g^{D, \mathcal{P}_{A_t}} : \{0, 1, \ldots, p\} \times \{0, 1, \ldots, 2k^2\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \to \{0, 1\}$ be a Boolean function, where $D = (\mathcal{P}_{\chi(t)} = (P_1, \ldots, P_p, O), \mathcal{Q}_{\chi(t)}, O) \in \mathcal{D}$ and $\mathcal{P}_{A_t} = (\tilde{P}_1, \ldots, \tilde{P}_{k'}) \in \mathcal{F}^{A_t}$.

1. **(Correctness)** The function $g^{D, \mathcal{P}_{A_t}}$ is *g-correct* if we have $g^{D, \mathcal{P}_{A_t}}(\ell, d, \bar{y}, q) = 1$ for all $\ell \in \{0, 1, \ldots, p\}$, $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{y} = (y_1, \ldots, y_k) \in \{0, 1, \ldots, \lambda\}^k$

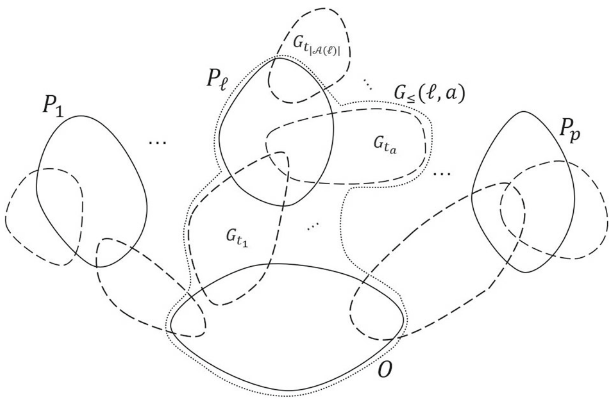

Fig. 3 An example of $G_{\leq}(\ell, a)$. Here the regions enclosed by dashed lines represent $V(G_{t'})$, where t' runs over $\mathcal{A}(\ell)$. The region enclosed by the dotted line is $G_{\leq}(\ell, a)$

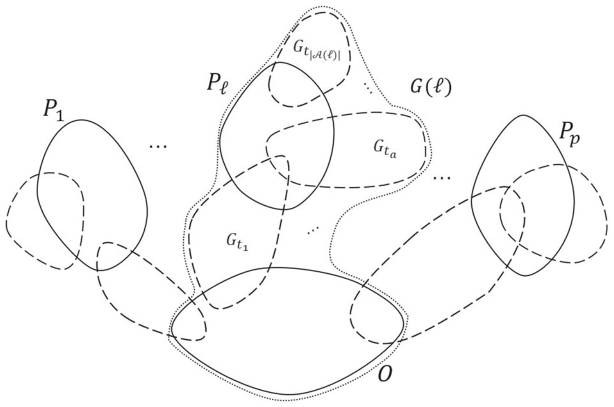

Fig. 4 An example of $G(\ell)$. Here the regions enclosed by dashed lines represent $V(G_{t'})$, where t' runs over children of t. The region enclosed by the dotted line is $G(\ell)$

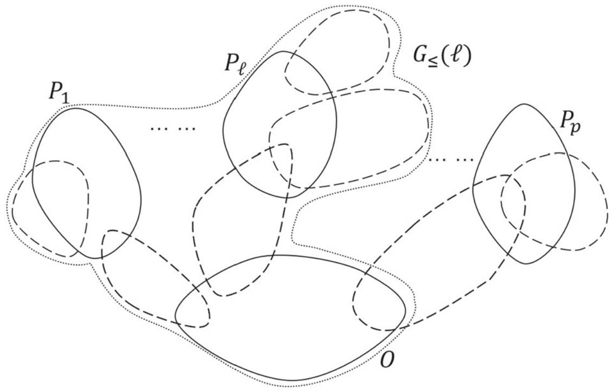

Fig. 5 An example of $G_{\le}(\ell)$. Here the regions enclosed by dashed lines represent $V(G_{t'})$, where t' runs over children of t. The region enclosed by the dotted line is $G_{\le}(\ell)$

and $q \in [k]$ for which there exists a k-subpartition $\mathcal{P} = (P'_1, \dots, P'_k)$ of $V(G_{\le}(\ell))$ satisfying the following conditions:

(i) $\mathcal{P}$ restricted to $O \cup P_{\le \ell}$ is a coarsening of $\mathcal{Q}_{\chi(t)}$ restricted to $O \cup P_{\le \ell}$,
(ii) $|\delta_{G_{\le}(\ell)}(P'_i)| = y_i$ for all $i \in [k]$,
(iii) $|\delta_T(\mathcal{P})| \le d$,
(iv) $O \subseteq P'_q$,
(v) if $A_t \subseteq O \cup P_{\le \ell}$, then $P'_i \cap A_t = \tilde{P}_i$, for all $i \in [k']$, and
(vi) $\mathcal{P}$ is a restriction of Ω to $V(G_{\le}(\ell))$.

A k-subpartition of $V(G_{\le}(\ell))$ satisfying the above six conditions is said to *witness* g-correctness of $g^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q)$.

2. **(Soundness)** The function $g^{D,\mathcal{P}_{A_t}}$ is g-*correct* if for all $\ell \in \{0, 1, \dots, p\}$, $d \in \{0, 1, \dots, 2k^2\}$, $\bar{y} = (y_1, \dots, y_k) \in \{0, 1, \dots, \lambda\}^k$ and $q \in [k]$, we have $g^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q) = 1$ only if there exists a k-subpartition $\mathcal{P} = (P'_1, \dots, P'_k)$ of

$V(G_{\le}(\ell))$ satisfying conditions (i), (ii), (iii), (iv) and (v) above. A k-subpartition of $V(G_{\le}(\ell))$ satisfying (i), (ii), (iii), (iv) and (v) is said to *witness* g-soundness of $g^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q)$.

Definition 12 Let $h^{D,\mathcal{P}_{A_t}} : \{0, 1, \ldots, p\} \times \{0, 1, \ldots, 2k^2\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \to \{0, 1\}$ be a Boolean function, where $D \in \mathcal{D}$ and $\mathcal{P}_{A_t} = (\tilde{P}_1, \ldots, \tilde{P}_{k'}) \in \mathcal{F}^{A_t}$.

1. **(Correctness)** The function h is *h-correct* if we have $h^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q) = 1$ for all $\ell \in \{0, 1, \ldots, p\}$, $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{y} = (y_1, \ldots, y_k) \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$ for which there exists a k-subpartition $\mathcal{P} = (P'_1, \ldots, P'_k)$ of $V(G(\ell))$ satisfying the following conditions:

   (i) $\mathcal{P}$ restricted to $O \cup P_\ell$ is a coarsening of $\mathcal{Q}_{\chi(t)}$ restricted to $O \cup P_\ell$,
   (ii) $|\delta_{G(\ell)}(P'_i)| = y_i$ for all $i \in [k]$,
   (iii) $|\delta_T(\mathcal{P})| \le d$,
   (iv) $O \subseteq P'_q$,
   (v) if $A_t \subseteq O \cup P_\ell$, then $P'_i \cap A_t = \tilde{P}_i$, for all $i \in [k']$, and
   (vi) $\mathcal{P}$ is a restriction of Ω to $V(G(\ell))$.

   A k-subpartition of $V(G(\ell))$ satisfying the above six conditions is said to *witness* h-correctness of $h^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q)$.
2. **(Soundness)** The function $h^{D,\mathcal{P}_{A_t}}$ is *h-correct* if for all $\ell \in \{0, 1, \ldots, p\}$, $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{y} = (y_1, \ldots, y_k) \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$, we have $h^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q) = 1$ only if there exists a k-subpartition $\mathcal{P} = (P'_1, \ldots, P'_k)$ of $V(G(\ell))$ satisfying conditions (i), (ii), (iii), (iv) and (v) above. Such k-subpartition $\mathcal{P}$ is said to *witness* h-soundness of $h^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q)$.

We note that the difference between $g^{D,\mathcal{P}_{A_t}}$ and $h^{D,\mathcal{P}_{A_t}}$ is that $g^{D,\mathcal{P}_{A_t}}$ considers $G_{\le}(\ell)$ and $O \cup P_{\le \ell}$, whereas $h^{D,\mathcal{P}_{A_t}}$ considers $G(\ell)$ and $O \cup P_\ell$. The following proposition outlines how to compute a function r^D that is both r-correct and r-sound using functions $g^{D,\mathcal{P}_{A_t}}$ for every $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$.

Proposition 3 *Suppose that we have functions* $g^{D,\mathcal{P}_{A_t}} : \{0, 1, \ldots, p\} \times \{0, 1, \ldots, 2k^2\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \to \{0, 1\}$ *for every* $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$ *such that all of them are both g-correct and g-sound. Then, the function* r^D *obtained by setting*

$$r^D(\mathcal{P}_{A_t}, \bar{x}, d) := \max\{g^{D,\mathcal{P}_{A_t}}(p, d, \bar{x}, q) : q \in [k]\}$$

for every $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, $\bar{x} \in \{0, 1, \ldots, \lambda\}^k$ *and* $d \in \{0, 1, \ldots, 2k^2\}$ *is both r-correct and r-sound.*

Proof We first show r-correctness. For $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, $\bar{x} \in \{0, 1, \ldots, \lambda\}^k$ and $d \in \{0, 1, \ldots, 2k^2\}$, suppose that there exists a k-subpartition $\Pi = (\pi_1, \ldots, \pi_k)$ of $V(G_t)$ witnessing r-correctness of $r^D(\mathcal{P}_{A_t}, \bar{x}, d)$. Then, there exists $q' \in [k]$ such that $O \subseteq \pi_{q'}$ by definition of D. It follows that Π also witnesses g-correctness of $g^{D,\mathcal{P}_{A_t}}(p, d, \bar{x}, q')$, and hence $g^{D,\mathcal{P}_{A_t}}(p, d, \bar{x}, q') = 1$ since the function $g^{D,\mathcal{P}_{A_t}}$ is g-correct. This implies that $r^D(\mathcal{P}_{A_t}, \bar{x}, d) = 1$.

Next, we show r-soundness. Suppose that $r^D(\mathcal{P}_{A_t}, \bar{x}, d) = 1$ for some $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, $\bar{x} \in \{0, 1, \ldots, \lambda\}^k$ and $d \in \{0, 1, \ldots, 2k^2\}$. Then, there exists $q' \in [k]$

such that $g^{D,\mathcal{P}_{A_t}}(p, d, \bar{x}, q') = 1$. By g-soundness of the function $g^{D,\mathcal{P}_{A_t}}$, there exists a k-subpartition Π' of $V(G_{\leq}(p)) = V(G_t)$ that witnesses g-soundness of $g^{D,\mathcal{P}_{A_t}}(p, d, \bar{x}, q')$. It follows that Π' also witnesses r-soundness of $r^D(\mathcal{P}_{A_t}, \bar{x}, d)$.
□

Our goal now is to compute a function $g^{D,\mathcal{P}_{A_t}}$ that is both g-correct and g-sound.

Lemma 8 *There exists an algorithm that takes as input (τ, χ), a tree node $t \in V(\tau)$, a partition $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, a nice decomposition $(\mathcal{P}_{\chi(t)} = (P_1, \dots, P_p, O), \mathcal{Q}_{\chi(t)}, O) \in \mathcal{D}$, together with Boolean functions $f_{t'} : \mathcal{F}^{A_{t'}} \times \{0, 1, \dots, \lambda\}^k \times \{0, 1, \dots, 2k^2\} \to \{0, 1\}$ for every child t' of t that are f-correct and f-sound, and runs in time $(\lambda k)^{O(k^2)} n^{O(1)}$ to return a function $g^{D,\mathcal{P}_{A_t}} : \{0, 1, \dots, p\} \times \{0, 1, \dots, 2k^2\} \times \{0, 1, \dots, \lambda\}^k \times [k] \to \{0, 1\}$ that is g-correct and g-sound.*

Lemma 7 follows from Lemma 8 and Proposition 3. We will prove Lemma 8 in the following subsections.

3.3.1 Computing $g^{D,\mathcal{P}_{A_t}}$ assuming $h^{D,\mathcal{P}_{A_t}}$ is available

In this section, for a given pair of $D = (\mathcal{P}_{\chi(t)} = (P_1, \dots, P_p, O), \mathcal{Q}_{\chi(t)}, O) \in \mathcal{D}$ and $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, we will show how to construct a function $g^{D,\mathcal{P}_{A_t}} : \{0, 1, \dots, p\} \times \{0, 1, \dots, 2k^2\} \times \{0, 1, \dots, \lambda\}^k \times [k] \to \{0, 1\}$ that is both g-correct and g-sound using a function $h^{D,\mathcal{P}_{A_t}} : \{0, 1, \dots, p\} \times \{0, 1, \dots, 2k^2\} \times \{0, 1, \dots, \lambda\}^k \times [k] \to \{0, 1\}$ that is h-correct and h-sound.

Lemma 9 *There exists an algorithm that takes as input a partition $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, a nice decomposition $D = (\mathcal{P}_{\chi(t)} = (P_1, \dots, P_p, O), \mathcal{Q}_{\chi(t)}, O) \in \mathcal{D}$, together with a Boolean function $h^{D,\mathcal{P}_{A_t}} : \{0, 1, \dots, p\} \times \{0, 1, \dots, 2k^2\} \times \{0, 1, \dots, \lambda\}^k \times [k] \to \{0, 1\}$ that is h-correct and h-sound, and runs in time $(\lambda k)^{O(k^2)} n^{O(1)}$ to return a function $g^{D,\mathcal{P}_{A_t}} : \{0, 1, \dots, p\} \times \{0, 1, \dots, 2k^2\} \times \{0, 1, \dots, \lambda\}^k \times [k] \to \{0, 1\}$ that is g-correct and g-sound.*

Proof Let the inputs be fixed as in the lemma. We will iteratively assign values to $g^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{w}, q)$ for $\ell = 0, 1, \dots, p$.

For $\ell = 0$, we set $g^{D,\mathcal{P}_{A_t}}(0, d, \bar{w}, q) := h^{D,\mathcal{P}_{A_t}}(0, d, \bar{w}, q)$ for every $d \in \{0, 1, \dots, 2k^2\}$, $\bar{w} \in \{0, 1, \dots, \lambda\}^k$ and $q \in [k]$. We note $O \cup P_{\leq 0} = O \cup P_0$ and $G_{\leq}(0) = G(0)$ by definition. Thus the definitions of g-correctness and h-correctness coincide, and the definitions of g-soundness and h-soundness coincide when $\ell = 0$.

Now, we will assume that for some $\ell \in [p]$, we have assigned values to $g^{D,\mathcal{P}_{A_t}}(\ell - 1, d, \bar{w}, q)$ for every $d \in \{0, 1, \dots, 2k^2\}$, $\bar{w} \in \{0, 1, \dots, \lambda\}^k$ and $q \in [k]$ and describe an algorithm to assign values to $g^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{w}, q)$ for every $d \in \{0, 1, \dots, 2k^2\}$, $\bar{w} \in \{0, 1, \dots, \lambda\}^k$ and $q \in [k]$.

Let

$$\begin{aligned}\mathcal{Y}^\ell := \big\{&((\ell - 1, d_1, \bar{y}, q_1), (\ell, d_2, \bar{z}, q_2)) :\\ &d_1, d_2 \in \{0, 1, \dots, 2k^2\}, \bar{y}, \bar{z} \in \{0, 1, \dots, \lambda\}^k, q_1, q_2 \in [k],\\ &g^{D,\mathcal{P}_{A_t}}(\ell - 1, d_1, \bar{y}, q_1) = 1, h^{D,\mathcal{P}_{A_t}}(\ell, d_2, \bar{z}, q_2) = 1\big\}.\end{aligned}$$

Here, $\mathcal{Y}^\ell$ is the collection of pairs of inputs to $g^{D,\mathcal{P}_{A_t}}$ and $h^{D,\mathcal{P}_{A_t}}$ such that $g^{D,\mathcal{P}_{A_t}}$ evaluates to 1 on the first input and $h^{D,\mathcal{P}_{A_t}}$ evaluates to 1 on the second input. For each pair $((\ell-1, d_1, \bar{y}, q_1), (\ell, d_2, \bar{z}, q_2)) \in \mathcal{Y}^\ell$ and each pair of permutations $\sigma_1, \sigma_2 : [k] \to [k]$ of permutations (i.e., bijections) satisfying the following conditions,

(G1) $\sigma_1(q_1) = \sigma_2(q_2)$,
(G2) If $A_t \subseteq O \cup P_\ell$, then $\sigma_2(i) = i$ for all $i \in [k']$.
(G3) If $A_t \subseteq O \cup P_{\leq \ell-1}$, then $\sigma_1(i) = i$ for all $i \in [k']$.

our algorithm will set $g^{D,\mathcal{P}_{A_t}}(\ell, d_3, \sigma_1(\bar{y}) + \sigma_2(\bar{z}), \sigma_1(q_1)) = 1$ for all $d_3 \geq d_1 + d_2$. Here the notation $\sigma_1(\bar{y})$ refers to the k-dimensional vector whose ith entry is $y_{\sigma_1^{-1}(i)}$. Finally, we set $g^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{w}, q) = 0$ for all $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{w} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$ for which the algorithm has not set the $g^{D,\mathcal{P}_{A_t}}$ value so far.

We briefly mention the intuition behind conditions (G1), (G2), and (G3). The goal of this algorithm is to merge a k-subpartition of $G_{\leq}(\ell-1)$ and a k-subpartition of $G(\ell)$ to get a k-subpartition of $G_{\leq}(\ell)$. Permutations σ_1 and σ_2 together allow us to merge the $\sigma_1(i)^{\text{th}}$ part of the k-subpartition of $G_{\leq}(\ell-1)$ with the $\sigma_2(i)^{\text{th}}$ part of the k-subpartition of $G(\ell)$ for each $i \in [k]$. However, the choice of σ_1 and σ_2 cannot be arbitrary. Condition (G1) guarantees that the parts containing O are merged together. Conditions (G3) guarantees condition (v) in Definition 11. Condition (G2) along with (G3) guarantees that when $A_t \subseteq O \cup P_\ell$, the two parts containing the same part of $\mathcal{P}_{A_t}$ from the two k-subpartitions are merged together. We describe this formally in Algorithm 1.

Algorithm 1 Computing function $g^{D,\mathcal{P}_{A_t}}$

```
Set $g^{D,\mathcal{P}_{A_t}}(0, d, \bar{w}, q) = h^{D,\mathcal{P}_{A_t}}(0, d, \bar{w}, q)$ for every $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{w} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$.
for all $\ell = 1, \ldots, p$ do
    for all pair $((\ell-1, d_1, \bar{y}, q_1), (\ell, d_2, \bar{z}, q_2)) \in \mathcal{Y}^\ell$ do
        for all permutation pairs $(\sigma_1, \sigma_2)$ satisfying (G1), (G2) and (G3) do
            Set $g^{D,\mathcal{P}_{A_t}}(\ell, d_3, \sigma_1(\bar{y}) + \sigma_2(\bar{z}), \sigma_1(q_1)) = 1$ for all $d_3 \geq d_1 + d_2$.
        end for
    end for
    For $d_3 \in \{0, 1, \ldots, 2k^2\}$, $\bar{w} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$ such that $g^{D,\mathcal{P}_{A_t}}(\ell, d_3, \bar{w}, q)$ is not yet set to 1, set $g^{D,\mathcal{P}_{A_t}}(\ell, d_3, \bar{w}, q) = 0$.
end for
```

We first bound the run-time of the algorithm. The size of $\mathcal{Y}^\ell$ is $\mathsf{O}(k^6)\lambda^{2k}$ for every $\ell \in [p]$. We recall that $p \leq n$. Thus, the run-time of the algorithm is

$$p \cdot \mathsf{O}(k^6)\lambda^{2k} \cdot (k!)^2 \mathsf{O}(k^2) = (\lambda k)^{\mathsf{O}(k^2)} n.$$

We now prove the correctness of the algorithm by induction on ℓ. We recall that we have already proved the base case. We now prove the induction step.

By induction hypothesis, if there exists a k-subpartition of $V(G_{\leq}(\ell-1))$ witnessing g-correctness of $g^{D,\mathcal{P}_{A_t}}(\ell-1, d, \bar{w}, q)$ for some $d \in \{0, 1, \ldots, 2k^2\}$,

$\bar{w} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$, then $g^{D,\mathcal{P}_{A_t}}(\ell-1, d, \bar{w}, q) = 1$. Furthermore, if $g^{D,\mathcal{P}_{A_t}}(\ell-1, d, \bar{w}, q) = 1$ for some $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{w} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$, then there exists a k-subpartition of $V(G_{\leq}(\ell-1))$ witnessing g-soundness of $g^{D,\mathcal{P}_{A_t}}(\ell-1, d, \bar{w}, q)$.

Suppose that there exists a k-subpartition $\mathcal{Q}^0 = (Q_1^0, \ldots, Q_k^0)$ of $V(G_{\leq}(\ell))$ witnessing g-correctness of $g^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{w}, q)$ for some $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{w} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$. We now show that the algorithm will correctly set $g^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{w}, q) = 1$.

Let $\mathcal{Q}^1 := (Q_1^1, \ldots, Q_k^1)$ and $\mathcal{Q}^2 := (Q_1^2, \ldots, Q_k^2)$ be restrictions of $\mathcal{Q}^0$ to the vertices of $G_{\leq}(\ell-1)$ and $G(\ell)$, respectively, given by $Q_i^1 := Q_i^0 \cap V(G_{\leq}(\ell-1))$ and $Q_i^2 := Q_i^0 \cap V(G(\ell))$ for every $i \in [k]$. It follows that $\mathcal{Q}^1$ and $\mathcal{Q}^2$ are both restrictions of Ω. We note that $\mathcal{Q}^1$ witnesses g-correctness of $g^{D,\mathcal{P}_{A_t}}(\ell-1, d_1, \bar{y}, q)$, where $d_1 := |\delta_T(\mathcal{Q}^1)|$ and $y_i := |\delta_{G_{\leq}(\ell-1)}(Q_i^1)|$ for all $i \in [k]$. Furthermore, $\mathcal{Q}^2$ witnesses h-correctness of $h^{D,\mathcal{P}_{A_t}}(\ell, d_2, \bar{z}, q)$ where $d_2 := |\delta_T(\mathcal{Q}^2)|$ and $z_i := |\delta_{G(\ell)}(Q_i^2)|$ for all $i \in [k]$. We note that $d_1, d_2 \in \{0, 1, \ldots, 2k^2\}$ and $\bar{y}, \bar{z} \in \{0, 1, \ldots, \lambda\}^k$ since the number of crossing edges in the subgraph is at most the number of crossing edges in the graph $G_{\leq}(\ell)$.

Consequently, the pair $((\ell-1, d_1, \bar{y}, q), (\ell, d_2, \bar{z}, q))$ is present in $\mathcal{Y}^\ell$. We now consider the case when σ_1 and σ_2 are both the identity permutation on $[k]$, i.e., $\sigma_1(i) = i = \sigma_2(i)$ for every $i \in [k]$. These two permutations satisfy the conditions (G1), (G2), and (G3). Moreover, since there are no edges between any two distinct parts among $P_1, \ldots, P_\ell$ due to the nice decomposition property, and that no two of $P_1, \ldots, P_\ell$ intersects the same adhesion of a child of t in τ (by property (iv) of Definition 9), we know that $d_1 + d_2 \leq d$ and $y_i + z_i = w_i$ for each $i \in [k]$. This also implies that $\sigma_1(\bar{y}) + \sigma_2(\bar{z}) = \bar{w}$. We also have that $\sigma_1(q) = q$. Consequently, the algorithm will set $g^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{w}, q)$ to be 1.

Next suppose that $((\ell-1, d_1, \bar{y}, q_1), (\ell, d_2, \bar{z}, q_2)) \in \mathcal{Y}^\ell$ and let $\sigma_1, \sigma_2 : [k] \to [k]$ be permutations satisfying conditions (G1), (G2), and (G3). Then, we will exhibit a k-subpartition $\mathcal{Q}^0$ of the vertices of $G_{\leq}(\ell)$ that witnesses g-soundness of $g^{D,\mathcal{P}_{A_t}}(\ell, d_3, \sigma_1(\bar{y}) + \sigma_2(\bar{z}), \sigma_1(q_1))$ for every $d_3 \geq d_1 + d_2$. Let $\mathcal{Q}^1 := (Q_1^1, \ldots, Q_k^1)$ and $\mathcal{Q}^2 := (Q_1^2, \ldots, Q_k^2)$ be k-subpartitions of the vertices of $G_{\leq}(\ell-1)$ and $G(\ell)$ respectively, that witness g-soundness of $g^{D,\mathcal{P}_{A_t}}(\ell-1, d_1, \bar{y}, q_1)$ and h-soundness of $h^{D,\mathcal{P}_{A_t}}(\ell, d_1, \bar{z}, q_2)$, respectively.

Consider the k-subpartition $\mathcal{Q}^0 := (Q_1^0, \ldots, Q_k^0)$ of the vertices of $G_{\leq}(\ell)$ obtained by setting $Q_i^0 := Q^1_{\sigma_1^{-1}(i)} \cup Q^2_{\sigma_2^{-1}(i)}$. Let $d_3 \geq d_1 + d_2$. We will show that $\mathcal{Q}^0$ witnesses g-soundness of $g^{D,\mathcal{P}_{A_t}}(\ell, d_3, \sigma_1(\bar{y}) + \sigma_2(\bar{z}), \sigma_1(q_1))$.

Since $\sigma_1(q_1) = \sigma_2(q_2)$, we know that the parts containing O in $\mathcal{Q}^1$ and $\mathcal{Q}^2$, i.e. $Q^1_{q_1}$ and $Q^2_{q_2}$, are both in $Q^0_{\sigma_1(q_1)}$, thus proving condition (iv) needed to witness g-soundness of $g^{D,\mathcal{P}_{A_t}}(\ell, d_3, \sigma_1(\bar{y}) + \sigma_2(\bar{z}), \sigma_1(q_1))$. As a consequence of the fact that $O \subseteq Q^1_{q_1}$ and $O \subseteq Q^2_{q_2}$, we obtain that $\mathcal{Q}^0$ is indeed a k-subpartition of the vertices of $G_{\leq}(\ell)$. Furthermore, the k-subpartition $\mathcal{Q}^0$ restricted to $O \cup P_{\leq \ell}$ is a coarsening of $\mathcal{Q}_{\chi(t)}$ restricted to $O \cup P_{\leq \ell}$, thus proving condition (i) needed to witness g-soundness of $g^{D,\mathcal{P}_{A_t}}(\ell, d_3, \sigma_1(\bar{y}) + \sigma_2(\bar{z}), \sigma_1(q_1))$. By the nice decomposition property, there are no edges between two distinct parts among $P_1, \ldots, P_\ell$. Hence,

$$\begin{aligned}|\delta_{G_{\leq}(\ell)}(Q_i^0)| &= |\delta_{G_{\leq}(\ell-1)}(Q^1_{\sigma_1^{-1}(i)})| + |\delta_{G(\ell)}(Q^2_{\sigma_2^{-1}(i)})| \\ &= y_{\sigma_1^{-1}(i)} + z_{\sigma_2^{-1}(i)} \text{ for all } i \in [k], \text{ and} \\ |\delta_T(\mathcal{Q}^0)| &= |\delta_T(\mathcal{Q}^1)| + |\delta_T(\mathcal{Q}^2)| \leq d_1 + d_2,\end{aligned}$$

thus proving conditions (ii) and (iii) needed to witness g-soundness of $g^{D,\mathcal{P}_{A_t}}(\ell, d_3, \sigma_1(\bar{y}) + \sigma_2(\bar{z}), \sigma_1(q_1))$.

Suppose that $A_t \subseteq O \cup P_\ell$. Then, we know that $\sigma_2(i) = i$ for all $i \in [k']$. We also know that $Q_i^2 \cap A_t = \tilde{P}_i$ for all $i \in [k']$. Therefore, $\tilde{P}_i \subseteq Q_i^2 \subseteq Q^0_{\sigma_2(i)} = Q_i^0$ for all $i \in [k']$. Next, suppose that $A_t \subseteq O \cup P_{\leq \ell}$. Then, we know that $\sigma_1(i) = i$ and $Q_i^1 \cap A_t = \tilde{P}_i$ for all $i \in [k']$. Therefore, $\tilde{P}_i \subseteq Q_i^1 \subseteq Q^0_{\sigma_1(i)} = Q_i^0$ for all $i \in [k']$. Hence, condition (v) needed to witness g-soundness of $g^{D,\mathcal{P}_{A_t}}(\ell, d_3, \sigma_1(\bar{y}) + \sigma_2(\bar{z}), \sigma_1(q_1))$ also holds. This shows that $\mathcal{Q}^0$ witnesses g-soundness of $g^{D,\mathcal{P}_{A_t}}(\ell, d_3, \sigma_1(\bar{y}) + \sigma_2(\bar{z}), \sigma_1(q_1))$ for all $d_3 \geq d_1 + d_2$. □

3.3.2 Computing $h^{D,\mathcal{P}_{A_t}}$ in leaf nodes of the tree decomposition

In this section we will describe an algorithm to compute an h-correct and h-sound function $h^{D,\mathcal{P}_{A_t}} : \{0, 1, \ldots, p\} \times \{0, 1, \ldots, 2k^2\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \to \{0, 1\}$ when $t \in V(\tau)$ is a leaf node of τ. This corresponds to the base case of our dynamic program. We note that this base case is also handled by our algorithm for computing $h^{D,\mathcal{P}_{A_t}}$ in non-leaf nodes given in Sect. 3.3.3. We include the base case explicitly as a precursor which will help the reader understand the non-leaf case in the next section.

Lemma 10 *If $t \in V(\tau)$ is a leaf node of τ, then there exists an algorithm that takes as input a partition $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, a nice decomposition $D = (\mathcal{P}_{\chi(t)} = (P_1, \ldots, P_p, O), \mathcal{Q}_{\chi(t)}, O) \in \mathcal{D}$, and runs in time $(\lambda k)^{\mathsf{O}(k^2)} n^{\mathsf{O}(1)}$ to return a function $h^{D,\mathcal{P}_{A_t}} : \{0, 1, \ldots, p\} \times \{0, 1, \ldots, 2k^2\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \to \{0, 1\}$ that is h-correct and h-sound.*

Proof Let the input be fixed as in the lemma. We will iteratively assign values to $h^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{w}, q)$ for $\ell = 0, 1, \ldots, p$.

Let $\ell \in \{0, 1, \ldots, p\}$. By the definition of nice decomposition, we know that the part P_ℓ contains $\mathsf{O}(k^2)$ parts of $\mathcal{Q}_{\chi(t)}$. Hence, $O \cup P_\ell$ contains $\mathsf{O}(k^2)$ parts from $\mathcal{Q}_{\chi(t)}$. Hence, we can enumerate all $k^{\mathsf{O}(k^2)}$ k-subpartitions of $O \cup P_\ell$ that coarsen $\mathcal{Q}_{\chi(t)}$ and explicitly verify if one of them satisfies the required conditions to witness h-soundness of $h^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q)$. If so, then we set the corresponding $h^{D,\mathcal{P}_{A_t}}(l, d, \bar{y}, q) = 1$ and otherwise set $h^{D,\mathcal{P}_{A_t}}(l, d, \bar{y}, q) = 0$. Thus, the time to compute $h^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q)$ for all $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{y} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$ is

$$(2k^2 + 1) \cdot \lambda^k \cdot k \cdot k^{\mathsf{O}(k^2)} n^{\mathsf{O}(1)} = (\lambda k)^{\mathsf{O}(k^2)} n^{\mathsf{O}(1)}.$$

In order to compute $h^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q)$ for all $\ell \in \{0, 1, \dots, p\}$, $d \in \{0, 1, \dots, 2k^2\}$, $\bar{y} \in \{0, 1, \dots, \lambda\}^k$ and $q \in [k]$, the total time required is $p(\lambda k)^{\mathsf{O}(k^2)} n^{\mathsf{O}(1)} = (\lambda k)^{\mathsf{O}(k^2)} n^{\mathsf{O}(1)}$, since $p \le n$. The resulting function $h^{D,\mathcal{P}_{A_t}}$ is h-sound as well as h-correct by construction. $\square$

3.3.3 Computing $h^{D,\mathcal{P}_{A_t}}$ in non-leaf nodes of the tree decomposition

In this section, we will describe an algorithm to compute a function $h^{D,\mathcal{P}_{A_t}} : \{0, 1, \dots, p\} \times \{0, 1, \dots, 2k^2\} \times \{0, 1, \dots, \lambda\}^k \times [k] \to \{0, 1\}$ that is h-correct and h-sound when t is a non-leaf node of the tree decomposition τ.

Lemma 11 *There exists an algorithm that takes as input (τ, χ), a non-leaf tree node $t \in V(\tau)$, a partition $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, a nice decomposition $D = (\mathcal{P}_{\chi(t)} = (P_1, \dots, P_p, O), \mathcal{Q}_{\chi(t)}, O) \in \mathcal{D}$, together with Boolean functions $f_{t'} : \mathcal{F}^{A_{t'}} \times \{0, 1, \dots, \lambda\}^k \times \{0, 1, \dots, 2k^2\} \to \{0, 1\}$ for each child t' of t that are f-correct and f-sound, and runs in time $(\lambda k)^{\mathsf{O}(k^2)} n^{\mathsf{O}(1)}$ to return a function $h^{D,\mathcal{P}_{A_t}} : \{0, 1, \dots, p\} \times \{0, 1, \dots, 2k^2\} \times \{0, 1, \dots, \lambda\}^k \times [k] \to \{0, 1\}$ that is h-correct and h-sound.*

Proof Given the inputs and an integer $\ell \in \{0, 1, \dots, p\}$, let us define $\tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$ to be the set of k-subpartitions $\mathcal{P} = (P'_1, \dots, P'_k)$ of $O \cup P_\ell$ that satisfy the following conditions:

(i) $\mathcal{P}$ coarsens $\mathcal{Q}_{\chi(t)}$ restricted to $O \cup P_\ell$,
(ii) $|\delta_T(\mathcal{P})| \le 2k^2$, and
(iii) if $A_t \subseteq O \cup P_\ell$, then $P'_i \cap A_t = \tilde{P}_i$ for all $i \in [k']$.

Since every k-subpartition in $\tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$ necessarily coarsens $\mathcal{Q}_{\chi(t)}$, we have the size bound $|\tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)| = k^{\mathsf{O}(k^2)}$ because $O \cup P_\ell$ contains $\mathsf{O}(k^2)$ parts of $\mathcal{Q}_{\chi(t)}$.

In order to compute an h-correct and h-sound function $h^{D,\mathcal{P}_{A_t}}$, we will employ a new sub-problem that we define below.

Definition 13 Let $\mu^{D,\mathcal{P}_{A_t},\ell} : \{0, 1, ..., 2k^2\} \times \{0, 1, \dots, \lambda\}^k \times [k] \times \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell) \to \{0, 1\}$ be a Boolean function, where $D = (\mathcal{P}_{\chi(t)} = (P_1, \dots, P_p, O), \mathcal{Q}_{\chi(t)}, O) \in \mathcal{D}$, $\mathcal{P}_{A_t} = (\tilde{P}_1, \dots, \tilde{P}_{k'}) \in \mathcal{F}^{A_t}$ and $\ell \in \{0, 1, \dots, p\}$.

1. **(Correctness)** The function $\mu^{D,\mathcal{P}_{A_t},\ell}$ is *μ-correct* if we have $\mu^{D,\mathcal{P}_{A_t},\ell}(d, \bar{y}, q, \mathcal{R}) = 1$ for all $d \in \{0, 1, \dots, 2k^2\}$, $\bar{y} = (y_1, \dots, y_k) \in \{0, 1, \dots, \lambda\}^k$, $q \in [k]$ and $\mathcal{R} \in \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$ for which there exists a k-subpartition $\mathcal{P} = (P'_1, \dots, P'_k)$ of $V(G(\ell))$ satisfying the following conditions:

   (i) $\mathcal{P}$ restricted to $O \cup P_\ell$ is $\mathcal{R}$,
   (ii) $|\delta_{G(\ell)}(P'_i)| = y_i$ for all $i \in [k]$,
   (iii) $|\delta_T(\mathcal{P})| \le d$,
   (iv) $O \subseteq P'_q$,
   (v) if $A_t \subseteq O \cup P_\ell$, then $P'_i \cap A_t = \tilde{P}_i$ for all $i \in [k']$, and
   (vi) $\mathcal{P}$ is a restriction of Ω to $V(G(\ell))$.

   A k-subpartition of $V(G(\ell))$ satisfying the above six conditions is said to *witness* μ-correctness of $\mu^{D,\mathcal{P}_{A_t},\ell}(d, \bar{y}, q, \mathcal{R})$.

2. **(Soundness)** The function $\mu^{D,\mathcal{P}_{A_t},\ell}$ is *μ-correct* if for all $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{y} = (y_1, \ldots, y_k) \in \{0, 1, \ldots, \lambda\}^k$, $q \in [k]$ and $\mathcal{R} \in \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$, we have $\mu^{D,\mathcal{P}_{A_t},\ell}(d, \bar{y}, q, \mathcal{R}) = 1$ only if there exists a k-subpartition $\mathcal{P} = (P'_1, \ldots, P'_k)$ of $V(G(\ell))$ satisfying conditions (i), (ii), (iii), (iv) and (v) above. A k-subpartition of $V(G(\ell))$ satisfying (i), (ii), (iii), (iv) and (v) is said to *witness* μ-soundness of $\mu^{D,\mathcal{P}_{A_t},\ell}(d, \bar{y}, q, \mathcal{R})$.

A function $\mu^{D,\mathcal{P}_{A_t},\ell}$ that is μ-correct and μ-sound helps compute a function $h^{D,\mathcal{P}_{A_t}}$ that is h-correct and h-sound by the following proposition.

Proposition 4 *Suppose that we have functions* $\mu^{D,\mathcal{P}_{A_t},\ell} : \{0, 1, ..., 2k^2\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \times \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell) \to \{0, 1\}$ *for all* $\ell \in \{0, 1, \ldots, p\}$ *such that all of them are μ-correct and μ-sound. Then the function* $h^{D,\mathcal{P}_{A_t}} : \{0, 1, \ldots, p\} \times \{0, 1, \ldots, 2k^2\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \to \{0, 1\}$ *obtained by setting*

$$h^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q) := \max\left\{\mu^{D,\mathcal{P}_{A_t},\ell}(d, \bar{y}, q, \mathcal{R}) : \mathcal{R} \in \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)\right\}$$

for every $\ell \in \{0, 1, \ldots, p\}$, $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{y} \in \{0, 1, \ldots, \lambda\}^k$ *and* $q \in [k]$ *is both h-correct and h-sound.*

Proof We first show h-correctness. For $\ell \in \{0, 1, \ldots, p\}$, $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{y} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$, suppose that there exists a k-subpartition $\Pi = (\pi_1, \ldots, \pi_k)$ of $V(G(\ell))$ withnessing h-correctness of $h^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q)$. Then, Π also witnesses μ-correctness of $\mu^{D,\mathcal{P}_{A_t},\ell}(d, \bar{y}, q, \Pi^0)$, where $\Pi^0 = (\pi_1 \cap (O \cup P_\ell), \ldots, \pi_k \cap (O \cup P_\ell)) \in \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$. We note that $\Pi^0 \in \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$ because the definition of $h^{D,\mathcal{P}_{A_t}}$ guarantees the three conditions of $\tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$. Since the function $\mu^{D,\mathcal{P}_{A_t},\ell}$ is μ-correct, we know that $\mu^{D,\mathcal{P}_{A_t},\ell}(d, \bar{y}, q, \Pi^0) = 1$. This implies that $h^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q) = 1$.

Next, we show h-soundness. Suppose that $h^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q) = 1$ for some $\ell \in \{0, 1, \ldots, p\}$, $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{y} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$. It then follows that there exists $\mathcal{R}' \in \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$ such that $\mu^{D,\mathcal{P}_{A_t},\ell}(d, \bar{y}, q, \mathcal{R}') = 1$. Since the function $\mu^{D,\mathcal{P}_{A_t},\ell}$ is μ-sound, we know that there exists some k-subpartition Π' of $V(G(\ell))$ witnessing μ-soundness of $\mu^{D,\mathcal{P}_{A_t},\ell}(d, \bar{y}, q, \mathcal{R}')$. It follows that Π' also witnesses h-soundness of $h^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q)$. □

By the above proposition, it suffices to assign values to $\mu^{D,\mathcal{P}_{A_t},\ell}(d, \bar{y}, q, \mathcal{R})$ for every $\ell \in \{0, 1, \ldots, p\}$, $\mathcal{R} \in \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$, $d \in \{0, 1, \ldots, 2k^2\}$ $\bar{y} \in \{0, 1, \ldots, \lambda\}^k$, and $q \in [k]$ so that the resulting function is μ-correct and μ-sound. We define another sub-problem.

Definition 14 Let $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}} : \{0, 1, \ldots, 2k^2\} \times \{0, 1, \ldots, |\mathcal{A}(\ell)|\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \to \{0, 1\}$ be a Boolean function, where $D = (\mathcal{P}_{\chi(t)} = (P_1, \ldots, P_p, O), \mathcal{Q}_{\chi(t)}, O) \in \mathcal{D}$, $\mathcal{P}_{A_t} = (\tilde{P}_1, \ldots, \tilde{P}_{k'}) \in \mathcal{F}^{A_t}$, $\ell \in \{0, 1, \ldots, p\}$ and $\mathcal{R} = (R_1, \ldots, R_k) \in \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$.

1. **(Correctness)** The function $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}$ is *ν-correct* if we have $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, a, \bar{z}, q) = 1$ for all $d \in \{0, 1, \ldots, 2k^2\}$, $a \in \{0, 1, ..., |\mathcal{A}(\ell)|\}$, $\bar{z} = (z_1, \ldots, z_k) \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$ for which there exists a k-subpartition $\mathcal{P} = (P'_1, \ldots, P'_k)$ of $V(G_{\le}(\ell, a))$ satisfying the following conditions:

(i) $P'_i \cap (O \cup P_\ell) = R_i$ for all $i \in [k]$,
(ii) $|\delta_{G_{\leq}(\ell,a)}(P'_i)| = z_i$ for all $i \in [k]$,
(iii) $|\delta_T(\mathcal{P})| \leq d$,
(iv) $O \subseteq P'_q$, and
(v) $\mathcal{P}$ is a restriction of Ω to $V(G_{\leq}(\ell, a))$.

A k-subpartition of $V(G_{\leq}(\ell, a))$ satisfying the above five conditions is said to *witness* ν-correctness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, a, \bar{z}, q)$.

2. **(Soundness)** The function $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}$ is ν-*sound* if for all $d \in \{0, 1, \ldots, 2k^2\}$, $a \in \{0, 1, ..., |\mathcal{A}(\ell)|\}$, $\bar{z} = (z_1, \ldots, z_k) \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$, we have $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, a, \bar{z}, q) = 1$ only if there exists a k-subpartition $\mathcal{P} = (P'_1, \ldots, P'_k)$ of $V(G_{\leq}(\ell))$ satisfying conditions (i), (ii), (iii) and (iv) above. A k-subpartition of $V(G_{\leq}(\ell, a))$ satisfying (i), (ii), (iii) and (iv) is said to *witness* ν-soundness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, a, \bar{z}, q)$.

A function $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}} : \{0, 1, \ldots, 2k^2\} \times \{0, 1, ..., |\mathcal{A}(\ell)|\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \to \{0, 1\}$ that is ν-correct and ν-sound helps compute a function $\mu^{D,\mathcal{P}_{A_t},\ell}$: $\{0, 1, ..., 2k^2\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \times \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell) \to \{0, 1\}$ that is μ-correct and μ-sound by the following observation. As a side note, we mention that setting $|\mathcal{A}(\ell)| = 0$ resolves the base case of t being a leaf node.

Observation 2 *Suppose that we have functions* $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}} : \{0, 1, \ldots, 2k^2\} \times \{0, 1, \ldots, |\mathcal{A}(\ell)|\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \to \{0, 1\}$ *for every* $\mathcal{R} \in \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$ *such that all of them are ν-correct and ν-sound. Then the function* $\mu^{D,\mathcal{P}_{A_t},\ell}$: $\{0, 1, ..., 2k^2\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \times \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell) \to \{0, 1\}$ *obtained by setting*

$$\mu^{D,\mathcal{P}_{A_t},\ell}(d, \bar{y}, q, \mathcal{R}) := \nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, |\mathcal{A}(\ell)|, \bar{y}, q)$$

for every $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{y} \in \{0, 1, \ldots, \lambda\}^k$, $q \in [k]$ *and* $\mathcal{R} \in \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$ *is both μ-correct and μ-sound.*

We note that a function $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}} : \{0, 1, \ldots, 2k^2\} \times \{0, 1, \ldots, |\mathcal{A}(\ell)|\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \to \{0, 1\}$ that is ν-correct and ν-sound can be computed in $(\lambda k)^{\mathsf{O}(k^2)}$ time using Lemma 12, which we state and prove after this proof.

Our algorithm to prove Lemma 11 starts by computing $\tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$ for every $\ell \in \{0, 1, \ldots, p\}$, which can be done in $k^{\mathsf{O}(k^2)}n$ time (since the size of $\tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$ is $k^{\mathsf{O}(k^2)}$ for every $\ell \in \{0, 1, \ldots, p\}$). For each $\ell \in \{0, 1, \ldots, p\}$ and $\mathcal{R} \in \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$, our algorithm assigns values to $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, |\mathcal{A}(\ell)|, \bar{y}, q)$ for all $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{y} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$ using Lemma 12. The algorithms uses these values to next assign values to $\mu^{D,\mathcal{P}_{A_t},\ell}(d, \bar{y}, q, \mathcal{R})$ for all $\ell \in \{0, 1, \ldots, p\}$, $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{y} \in \{0, 1, \ldots, \lambda\}^k$, $q \in [k]$ and $\mathcal{R} \in \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$ using Observation 2. Finally, the algorithm uses these values to assign values to $h^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q)$ for all $\ell \in \{0, 1, \ldots, p\}$, $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{y} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$ using Proposition 4. The resulting function $h^{D,\mathcal{P}_{A_t}}$ is h-correct and h-sound.

The total time to compute $h^{D,\mathcal{P}_{A_t}}(\ell, d, \bar{y}, q)$ for all $\ell \in \{0, 1, \ldots, p\}$, $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{y} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$ is

$$(\lambda k)^{\mathsf{O}(k^2)}n^{\mathsf{O}(1)} + (p+1) \cdot |\tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)| \cdot (\lambda k)^{\mathsf{O}(k^2)}n^{\mathsf{O}(1)} = (\lambda k)^{\mathsf{O}(k^2)}n^{\mathsf{O}(1)}.$$

This completes the Proof of Lemma 11. □

Lemma 12 *There exists an algorithm that takes as input (τ, χ), a non-leaf tree node $t \in V(\tau)$, a partition $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, a nice decomposition $D = (\mathcal{P}_{\chi(t)} = (P_1, \ldots, P_p, O), \mathcal{Q}_{\chi(t)}, O) \in \mathcal{D}$, an integer $\ell \in \{0, 1, \ldots, p\}$, a k-subpartition $\mathcal{R} = (R_1, \ldots, R_k) \in \tilde{\mathcal{R}}(D, \mathcal{P}_{A_t}, \ell)$, together with Boolean functions $f_{t'} : \mathcal{F}^{A_{t'}} \times \{0, 1, \ldots, \lambda\}^k \times \{0, 1, \ldots, 2k^2\} \to \{0, 1\}$ for each child t' of t that are f-correct and f-sound, and runs in $(\lambda k)^{O(k^2)} n^{O(1)}$ time to return a function $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}} : \{0, 1, \ldots, 2k^2\} \times \{0, 1, ..., |\mathcal{A}(\ell)|\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \to \{0, 1\}$ that is ν-correct and ν-sound.*

Proof Let the input be as stated in the lemma. We will iteratively assign values to $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, a, \bar{z}, q)$ for $a = 0, 1, \ldots, p$.

For $a = 0$, we observe that in order to assign values to $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, 0, \bar{z}, q)$ for all $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{z} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$, the k-subpartition $\mathcal{R}$ is the only k-subpartition of $V(G_{\leq}(\ell, 0)) = O \cup P_l$ whose restriction to $O \cup P_\ell$ is $\mathcal{R}$ (i.e., it is the only k-subpartition that can satisfy condition (i) in the definition of the $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}$ subproblem). Therefore, we set $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, 0, \bar{z}, q) = 1$ if and only if $\mathcal{R}$ satisfies the remaining ν-soundness conditions, which can be verified in $n^{O(1)}$ time. The run-time to assign values to $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, 0, \bar{z}, q)$ for all $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{z} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$ is $\lambda^{O(k)} k^{O(1)} n^{O(1)}$. If $\mathcal{R}$ witnesses ν-correctness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, 0, \bar{z}, q)$ for some $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{z} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$, then our algorithm sets $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, 0, \bar{z}, q) = 1$. If our algorithm sets $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, 0, \bar{z}, q) = 1$ for some $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{z} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$, then $\mathcal{R}$ witnesses ν-soundness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, 0, \bar{z}, q)$.

Now, we will assume that for some $a \in [|\mathcal{A}(\ell)|]$, we have assigned values to $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, a-1, \bar{z}, q)$ for all $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{z} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$ and describe an algorithm to assign values to $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, a, \bar{z}, q)$ for all $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{z} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$.

Let $\mathcal{R}_{A_{t_a}} = (R'_1, ..., R'_{k_a})$ be a restriction of $\mathcal{R}$ to A_{t_a}, where $k_a \leq k$ and $R'_i \neq \emptyset$ for all $i \in [k_a]$ (i.e., $\mathcal{R}_{A_{t_a}}$ is a partition with at most k parts). Additionally, in the case $t_a = t$, we order $\mathcal{R}_{A_{t_a}}$ so that $\mathcal{R}_{A_{t_a}} = \mathcal{P}_{A_t}$. Moreover, let $\gamma : [k_a] \to [k]$ be the injection such that $R'_i = R_{\gamma(i)} \cap A_{t_a}$.

We start by defining a set

$$\begin{aligned}\mathcal{Z}^a := \Big\{&((d_1, a-1, \bar{z}, q), (\mathcal{R}_{A_{t_a}}, \bar{x}, d_2)) : \\ &d_1, d_2 \in \{0, 1, \ldots, 2k^2\}, \bar{z}, \bar{x} \in \{0, 1, \ldots, \lambda\}^k, q \in [k], \\ &\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d_1, a-1, \bar{z}, q) = 1,\ f_{A_{t_a}}(\mathcal{R}_{A_{t_a}}, \bar{x}, d_2) = 1\Big\}.\end{aligned}$$

For each pair $((d_1, a-1, \bar{z}, q), (\mathcal{R}_{A_{t_a}}, \bar{x}, d_2)) \in \mathcal{Z}^a$, and each pair of permutations $(\sigma_1, \sigma_2) : [k] \to [k]$ satisfying the following two conditions,

(N1) $\sigma_1(i) = \sigma_2(\gamma^{-1}(i))$ for all $i \in [k]$ with $R_i \cap A_{t_a} \neq \emptyset$, and
(N2) $\sigma_1(i) = i$ for all $i \in [k]$ for which $R_i \neq \emptyset$,

our algorithm will set $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d_3, a, \bar{w}, \sigma_1(q)) = 1$ for all $d_3 \geq d_1 + d_2 - |\delta_T(\mathcal{R}_{A_{t_a}})|$, where

$$w_i := z_{\sigma_1^{-1}(i)} + x_{\sigma_2^{-1}(i)} - \mathbb{1}[\sigma_2^{-1}(i) \leq k_a] \cdot |\delta_{G[A_{t_a}]}(R'_{\sigma_2^{-1}(i)})|$$

for all $i \in [k]$. We mention the informal intuition underlying conditions (N1) and (N2). The goal of this algorithm is to merge a k-subpartition of $V(G_{\leq}(\ell, a-1))$ and a k-subpartition of $V(G_{t_a})$ to construct a k-subpartition of $V(G_{\leq}(\ell, a))$. Permutations σ_1 and σ_2 allows us to merge the $\sigma_1(i)^{\text{th}}$ part of the first k-subpartition with the $\sigma_2(i)^{\text{th}}$ part of the second k-subpartition for every $i \in [k]$. However, the choice of σ_1 and σ_2 cannot be arbitrary: Condition (N1) guarantees that parts containing the same $R_i \cap A_{t_a}$ are merged together for each $i \in [k]$, and condition (N2) guarantees condition (i) in the definition of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}$.

Finally, we set $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d', a, \bar{w}, q) = 0$ for all $d' \in \{0, 1, \ldots, 2k^2\}$, $\bar{w} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$ for which the algorithm has not set the $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}$ value so far. This algorithm is described in Algorithm 2.

Algorithm 2 Computing function $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}$

```
for all $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{z} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$ do
    if $\mathcal{R}$ witnesses $\nu$-soundness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, 0, \bar{z}, q)$ then
        Set $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, 0, \bar{z}, q) = 1$.
    else
        Set $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, 0, \bar{z}, q) = 0$.
    end if
end for
for all $a = 1, 2, \ldots, |\mathcal{A}(\ell)|$ do
    for all pair $((d_1, a-1, \bar{z}, q), (\mathcal{R}_{A_{t_a}}, \bar{x}, d_2)) \in \mathcal{Z}^a$ do
        for all $(\sigma_1, \sigma_2)$ satisfying (N1) and (N2) do
            Set $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d_3, a, \bar{w}, \sigma_1(q)) = 1$ for all $d_3 \geq d_1 + d_2 - |\delta_T(\mathcal{R}_{A_{t_a}})|$,
            where $w_i := z_{\sigma_1^{-1}(i)} + x_{\sigma_2^{-1}(i)} - \mathbb{1}[\sigma_2^{-1}(i) \leq k_a] \cdot |\delta_{G[A_{t_a}]}(R'_{\sigma_2^{-1}(i)})|$ for all $i \in [k]$.
        end for
    end for
    For all $d' \in \{0, 1, \ldots, 2k^2\}$, $\bar{w} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$ with $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d', a, \bar{w}, q)$ not yet set
    to 1, set $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d', a, \bar{w}, q) = 0$.
end for
```

We first bound the run-time of the algorithm. The size of $\mathcal{Z}^a$ is $\mathsf{O}(k^5)\lambda^{2k}$ for every $a \in [|\mathcal{A}(\ell)|]$. We note that $|\mathcal{A}(\ell)| \leq |V(\tau)| = \text{poly}(n, \lambda, k)$. Thus, the run-time of the algorithm is

$$|\mathcal{A}(\ell)| \cdot \mathsf{O}(k^5)\lambda^{2k} \cdot (k!)^2 \mathsf{O}(k^2) = (\lambda k)^{\mathsf{O}(k^2)} n^{\mathsf{O}(1)}.$$

We now prove the correctness of Algorithm 2. We recall that we have already proved the base case. We now prove the induction step.

By induction hypothesis, if there exists a k-subpartition of $V(G_{\leq}(\ell, a-1))$ witnessing ν-correctness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, a-1, \bar{w}, q)$ for some $d \in \{0, 1, \ldots, 2k^2\}$,

$\bar{w} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$, then $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, a-1, \bar{w}, q) = 1$. Furthermore, if $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, a-1, \bar{w}, q) = 1$ for some $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{w} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$, then there exists a k-subpartition of $V(G_{\le}(\ell, a-1))$ witnessing ν-soundness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, a-1, \bar{w}, q)$.

Suppose that there exists a k-subpartition $\mathcal{Q}^0 = (Q_1^0, \ldots, Q_k^0)$ of $G_{\le}(\ell, a)$ that witnesses ν-correctness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, a, \bar{w}, q)$ for some $d \in \{0, 1, \ldots, 2k^2\}$, $\bar{w} \in \{0, 1, \ldots, \lambda\}^k$ and $q \in [k]$. We now show that the algorithm will correctly set $\nu(d, a, \bar{w}, q) = 1$.

Let $\mathcal{Q}^1 = (Q_1^1, \ldots, Q_k^1)$ be the restriction of $\mathcal{Q}^0$ to $V(G_{\le}(\ell, a))$ given by $Q_i^1 := Q_i^0 \cap V(G_{\le}(\ell, a-1))$ for all $i \in [k]$. Let $\mathcal{Q}^2 = (Q_1^2, \ldots, Q_k^2)$ be the restriction of $\mathcal{Q}^0$ to $V(G_{t_a})$ given by $Q_i^2 \cap A_{t_a} = R_i'$ for all $i \in [k_a]$. It follows that $\mathcal{Q}^1$ and $\mathcal{Q}^2$ are both restrictions of Ω. Let $\theta : [k] \to [k]$ be the permutation of $[k]$ such that $Q_i^2 = Q_{\theta(i)}^0 \cap V(G_{t_a})$ for every $i \in [k]$.

We note that $\mathcal{Q}^1$ witnesses ν-correctness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d_1, a-1, \bar{z}, q) = 1$, where $d_1 := |\delta_T(\mathcal{Q}^1)|$ and $z_i := |\delta_{G_{\le(\ell,a-1)}}(Q_i^1)|$ for every $i \in [k]$. Furthermore, $\mathcal{Q}^2$ witnesses ν-correctness of $f_{t_a}(\mathcal{R}_{A_{t_a}}, \bar{x}, d_2) = 1$, where $x_i := |\delta_{G_{t_a}}(Q_i^2)|$ for all $i \in [k]$ and $d_2 := |\delta_T(\mathcal{Q}^2)|$. We note that $d_1, d_2 \in \{0, 1, \ldots, 2k^2\}$ and $\bar{z}, \bar{x} \in \{0, 1, \ldots, \lambda\}^k$ since the number of crossing edges in the corresponding subgraph is at most the number of crossing edges in the graph $G_{\le}(\ell, a)$.

Hence, the pair $((d_1, a-1, \bar{z}, q), (\mathcal{R}_{A_{t_a}}, \bar{x}, d_2))$ is present in $\mathcal{Z}^a$. Consider the pair of permutations (σ_1, θ), where σ_1 is the identity permutation on $[k]$, i.e. $\sigma_1(i) = i$ for every $i \in [k]$.

We will first prove that (σ_1, θ) is an eligible pair of permutations for the algorithm. We note that by definition of θ, for all $i \in [k]$ such that $R_i \cap A_{t_a} \ne \emptyset$, the part Q_i^0 consists of $Q_{\theta^{-1}(i)}^2$ and Q_i^1, and Q_i^1 further contains R_i and $R'_{\gamma^{-1}(i)}$. Since each part of $\mathcal{Q}^0$ intersects at most one part of $\mathcal{R}_{A_{t_a}}$, we know that $R'_{\gamma^{-1}(i)} = Q_i^0 \cap V(G_{t_a}) = Q_{\theta^{-1}(i)}^2$. By definition of $\mathcal{Q}^2$, we know that $\gamma^{-1}(i) = \theta^{-1}(i)$. This implies that $\theta(\gamma^{-1}(i)) = i = \sigma_1(i)$ for all $i \in [k]$ such that $R_i \cap A_{t_a} \ne \emptyset$, which proves (N1). Since σ_1 is the identity permutation, it also satisfies (N2).

For this choice of permutations, we will show that our algorithm will set $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, a, \bar{w}, q) = 1$. By compactness of the tree decomposition (τ, χ), there are no edges between any two distinct members among $V(G_{t_1}) \backslash A_{t_1}, \ldots, V(G_{t_a}) \backslash A_{t_a}$. Consequently, our algorithm indeed sets $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d, a, \bar{w}, q) = 1$ due to the following relationships:

$$\begin{aligned} d &\ge |\delta_T(\mathcal{Q}^0)| = d_1 + d_2 - |\delta_T(\mathcal{R}_{A_{t_a}})|, \text{ and} \\ w_i &= |\delta_{G_{\le(\ell,a)}}(Q_i^0)| \\ &= |\delta_{G_{\le(\ell,a-1)}}(Q_i^1)| + |\delta_{G_{t_a}}(Q_{\theta^{-1}(i)}^2)| - \mathbb{1}[\theta^{-1}(i) \le k_a] \cdot |\delta_{G[A_{t_a}]}(R'_{\theta^{-1}(i)})| \\ &= z_i + x_{\theta^{-1}(i)} - \mathbb{1}[\theta^{-1}(i) \le k_a] \cdot |\delta_{G[A_{t_a}]}(R'_{\theta^{-1}(i)})| \quad \text{for all } i \in [k]. \end{aligned}$$

Next suppose that $((d_1, a-1, \bar{z}, q), (\mathcal{R}_{A_{t_a}}, \bar{x}, d_2)) \in \mathcal{Z}^a$ and let (σ_1, σ_2) be a pair of permutations satisfying (N1) and (N2). We will exhibit a k-subpartition $\mathcal{Q}^0$ of $V(G_{\le}(\ell, a))$ that witnesses ν-soundness of $\nu(d_3, a, \bar{w}, \sigma_1(q)) = 1$ for all $d_3 \ge$

$d_1 + d_2$ and $\bar{w}$ as described above. Let $\mathcal{Q}^1 = (Q_1^1, \ldots, Q_k^1)$ and $\mathcal{Q}^2 = (Q_1^2, \ldots, Q_k^2)$ be k-subpartitions of $V(G_{\leq}(\ell, a-1))$ and $V(G_{t_a})$ that witnesses ν-soundness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d_1, a-1, \bar{z}, q)$ and f-soundness of $f_{t_a}(\mathcal{R}_{A_{t_a}}, \bar{x}, d_2)$, respectively. Then, by definition, we have $Q_i^2 \cap A_{t_a} = R_i'$ for all $i \in [k_a]$.

Consider the k-subpartition $\mathcal{Q}^0 := (Q_1^0, \ldots, Q_k^0)$ of $V(G_{\leq}(\ell, a))$ obtained by setting $Q_i^0 := Q^1_{\sigma_1^{-1}(i)} \cup Q^2_{\sigma_2^{-1}(i)}$. Let $d_3 \geq d_1 + d_2$ and $\bar{w}$ be as described above. We will show that $\mathcal{Q}^0$ witnesses ν-soundness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d_3, a, \bar{w}, \sigma_1(q))$ by showing that it satisfies the five conditions in the definition of ν-soundness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d_3, a, \bar{w}, \sigma_1(q))$.

We first show that $\mathcal{Q}^0$ is indeed a k-subpartition of $V(G_{\leq}(\ell, a))$. It suffices to show that if a part R_i of $\mathcal{R}$ has non-empty intersection with A_{t_a}, then the part containing R_i in $\mathcal{Q}^1$, i.e., the part Q_i^1, is in the same part in $\mathcal{Q}^0$ with the part containing $R_i \cap A_{t_a} = R'_{\gamma^{-1}(i)}$ in $\mathcal{Q}^2$, i.e. the part $Q^2_{\gamma^{-1}(i)}$. This is equivalent to requiring $\sigma_1(i) = \sigma_2(\gamma^{-1}(i))$ for all $i \in [k]$ with $R_i \cap A_{t_a} \neq \emptyset$, which is satisfied due to condition (N1). Additionally, for all $i \in [k]$ such that $R_i \neq \emptyset$, we have that $Q_i^0 \cap (O \cup P_\ell) = Q^1_{\sigma_1^{-1}(i)} \cap (O \cup P_\ell) = R_i$ due to (N2). For all $i \in [k]$ such that $R_i = \emptyset$, we have $Q_i^0 \cap (O \cup P_\ell) = Q^1_{\sigma_1^{-1}(i)} \cap (O \cup P_\ell) = \emptyset = R_i$, where $Q^1_{\sigma_1^{-1}(i)} \cap (O \cup P_\ell) = \emptyset$ follows from condition (i) of $\mathcal{Q}^1$ witnessing ν-soundness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d_1, a-1, \bar{z}, q)$. Hence, this implies condition (i) needed to witness ν-soundness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d_3, a, \bar{w}, \sigma_1(q))$.

By compactness of the tree decomposition (τ, χ), we know that there is no edge between any two distinct members of $V(G_{t_1})\backslash A_{t_1}, \ldots, V(G_{t_a})\backslash A_{t_a}$. This implies that for a part Q_i^0 of $\mathcal{Q}^0$, if $Q^2_{\sigma_2^{-1}(i)}$, which is contained in Q_i^0, does not intersect A_{t_a}, then

$$|\delta_{G_{\leq}(\ell,a)}(Q_i^0)| = |\delta_{G_{\leq}(\ell,a-1)}(Q^0_{\sigma_1^{-1}(i)})| + |\delta_{G_{t_a}}(Q^0_{\sigma_2^{-1}(i)})| = z_{\sigma_1^{-1}(i)} + x_{\sigma_2^{-1}(i)}.$$

If $Q^2_{\sigma_2^{-1}(i)}$ intersects A_{t_a}, then

$$|\delta_{G_{\leq}(\ell,a)}(Q_i^0)| = z_{\sigma_1^{-1}(i)} + x_{\sigma_2^{-1}(i)} - |\delta_{G[A_{t_a}]}(R'_{\sigma_2^{-1}(i)})|.$$

Since $Q^2_{\sigma_2^{-1}(i)}$ intersects A_{t_a} if and only if $\sigma_2^{-1}(i) \leq k_a$, combining these two equations, we get

$$|\delta_{G_{\leq}(\ell,a)}(Q_i^0)| = z_{\sigma_1^{-1}(i)} + x_{\sigma_2^{-1}(i)} - \mathbb{1}[\sigma_2^{-1}(i) \leq k_a] \cdot |\delta_{G[A_{t_a}]}(R'_{\sigma_2^{-1}(i)})| = w_i.$$

The above holds for every $i \in [k]$, thus implying condition (ii) needed to witness ν-soundness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d_3, a, \bar{w}, \sigma_1(q))$. The part in $\mathcal{Q}^0$ that contains O is $Q^0_{\sigma_1(q)}$, thus implying condition (iv) needed to witness ν-soundness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d_3, a, \bar{w}, \sigma_1(q))$.

By definition of the tree decomposition (τ, χ), we know that there are no edges between $V(G_{t_a})\backslash A_{t_a}$ and $V(G_{\leq}(\ell, a-1))\backslash A_{t_a}$. This implies that

$$|\delta_T(\mathcal{Q}^0)| = |\delta_T(\mathcal{Q}^1)| + |\delta_T(\mathcal{Q}^2)| - |\delta_T(\mathcal{R}_{A_{t_a}})| \leq d_1 + d_2 - |\delta_T(\mathcal{R}_{A_{t_a}})|,$$

thus implying condition (iii) needed to witness ν-soundness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d_3, a, \bar{w}, \sigma_1(q))$. This shows that $\mathcal{Q}^0$ indeed witnesses ν-soundness of $\nu^{D,\mathcal{P}_{A_t},\ell,\mathcal{R}}(d_3, a, \bar{w}, \sigma_1(q)) = 1$, where d_3 and $\bar{w}$ are in the range given in our algorithm. □

3.3.4 Proof of Lemma 8

In this subsection, we complete the Proof of Lemma 8.

Proof of Lemma 8 Let the inputs be as stated in Lemma 8. First we will consider the case where $t \in V(\tau)$ is a leaf node. By Lemma 10, we can compute $h^{D,\mathcal{P}_{A_t}} : \{0, 1, \ldots, p\} \times \{0, 1, \ldots, 2k^2\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \to \{0, 1\}$ that is h-correct and h-sound in time $(\lambda k)^{O(k^2)} n^{O(1)}$.

If $t \in V(\tau)$ is not a leaf node, then by Lemma 11, we can compute a function $h^{D,\mathcal{P}_{A_t}} : \{0, 1, \ldots, p\} \times \{0, 1, \ldots, 2k^2\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \to \{0, 1\}$ that is h-correct and h-sound in $(\lambda k)^{O(k^2)} n^{O(1)}$ time.

Therefore, in either case, to compute a desired function $h^{D,\mathcal{P}_{A_t}} : \{0, 1, \ldots, p\} \times \{0, 1, \ldots, 2k^2\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \to \{0, 1\}$ that is h-correct and h-sound takes total time $(\lambda k)^{O(k^2)} n^{O(1)}$. By Lemma 9, we can compute a function $g^{D,\mathcal{P}_{A_t}} : \{0, 1, \ldots, p\} \times \{0, 1, \ldots, 2k^2\} \times \{0, 1, \ldots, \lambda\}^k \times [k] \to \{0, 1\}$ that is g-correct and g-sound with an additional $(\lambda k)^{O(k^2)} n$ time. This completes the Proof of Lemma 8. □

3.4 Generating nice decompositions and Proof of Lemma 6

In this section, we restate and prove Lemma 6.

Lemma 6 *There exists an algorithm that takes as input the spanning tree T, the tree decomposition (τ, χ), a tree node $t \in V(\tau)$, and runs in time $(\lambda k)^{O(k^2)} n^{O(1)}$ to return a family $\mathcal{D}$ of nice decompositions of $\chi(t)$ with $|\mathcal{D}| = (\lambda k)^{O(k^2)} n^{O(1)}$. Additionally, if a k-subpartition Π of $V(G_t)$ realizes f-correctness of $f_t(\mathcal{P}_{A_t}, \bar{x}, d)$ for some $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, $\bar{x} \in \{0, 1, \ldots, \lambda\}^k$ and $d \in \{0, 1, \ldots, 2k^2\}$, then $\mathcal{D}$ contains a nice decomposition $(\mathcal{P}_{\chi(t)}, \mathcal{Q}_{\chi(t)}, O)$ where $\mathcal{Q}_{\chi(t)}$ refines a restriction of Π to $\chi(t)$.*

Our definition of nice decomposition closely resembles the definition of [23]. Our way to generate nice decompositions and thereby prove Lemma 6 will also closely resemble the proof approach of [23]. We need the following lemma.

Lemma 13 (Lemma 2.1 of [23]) *There exists an algorithm that takes as input a set S and positive integers $s_1, s_2 \leq |S|$, and runs in time $O((s_1 + s_2)^{O(s_1)} |S|^{O(1)})$ to return a family $\mathcal{S} \subseteq 2^S$ of size $O((s_1 + s_2)^{O(s_1)} \log |S|)$ such that for every pair of disjoint subsets $X_1, X_2 \subseteq S$ where $|X_1| \leq s_1$ and $|X_2| \leq s_2$, there exists a set $X \in \mathcal{S}$ with $X_1 \subseteq X \subseteq S \backslash X_2$.*

Let the inputs be as stated in Lemma 6. We will start with notations followed by the algorithm with bound on the runtime and a proof of correctness.

Notations. We refer the reader to Definition 7 for the definition of $\text{proj}(T, \chi(t))$. Let $A \subseteq E(\text{proj}(T, \chi(t)))$. We let $\mathcal{R}_A$ denote the partition of $V(\text{proj}(T, \chi(t)))$ whose parts are the connected components of $\text{proj}(T, \chi(t)) \setminus A$. Let P and P' be disjoint subsets of vertices of $\text{proj}(T, \chi(t))$. We say that P *shares adhesion* with P' if there exists some descendant t' of t (inclusive) such that $P \cap A_{t'} \neq \emptyset$ and $P' \cap A_{t'} \neq \emptyset$. When it is necessary to specify which adhesion is shared by P and P', we will say that P *shares adhesion with* P' *via* $A_{t'}$ if t' is a descendant of t (inclusive) such that $A_{t'}$ intersects both P and P'. Moreover, we say that P *shares* G_t*-edge* with P' if there is an edge in G_t with one end-vertex in P and another end-vertex in P'. For a partition $\mathcal{R}$ of $V(\text{proj}(T, \chi(t)))$, we define the graph $H(\mathcal{R})$ whose vertices correspond to the parts of $\mathcal{R}$ and two vertices are adjacent in $H(\mathcal{R})$ if and only if the corresponding parts in $\mathcal{R}$ share either an adhesion or a G_t-edge with each other.

Algorithm. We now describe the algorithm. We initialize $\mathcal{D}$ to be the empty set. The first step of our algorithm is to generate a family $\mathcal{C}$ using Lemma 13 on set $E(\text{proj}(T, \chi(t)))$ with $s_1 = \min\{|E(\text{proj}(T, \chi(t)))|, 4k^2\}$ and $s_2 = \min\{|E(\text{proj}(T, \chi(t)))|, (4k^2+1) \cdot 2((\lambda k + 1)^5 + 4k^2 + 1)\}$. The next step of our algorithm depends on the size of the bag $\chi(t)$.

Case 1: Suppose $|\chi(t)| \leq (\lambda k + 1)^5$. For each $C' \in \mathcal{C}$, if $\mathcal{R}_{C'}$ has at most $4k^2 + 1$ parts, then we add the following triple to $\mathcal{D}$:

$$D_{C'} := \left(\mathcal{P}_{\chi(t)} := (\chi(t)), \mathcal{Q}_{\chi(t)} := \mathcal{R}_{C'}|_{\chi(t)}, O := \emptyset\right),$$

where the partition $\mathcal{R}_{C'}|_{\chi(t)}$ is a restriction of $\mathcal{R}_{C'}$ to $\chi(t)$. If $\mathcal{R}_{C'}$ has more than $4k^2 + 1$ parts, then we do not add anything into $\mathcal{D}$.

Case 2: Suppose $|\chi(t)| > (\lambda k + 1)^5$. For each $C' \in \mathcal{C}$, we generate a family $\mathcal{S}_{C'}$ using Lemma 13 on the set $\{P : P \text{ is a part of } \mathcal{R}_{C'}\}$ with $s_1 = \min\{|\mathcal{R}_{C'}|, 4k^2 + 1\}$ and $s_2 = \min\{|\mathcal{R}_{C'}|, (4k^2 + 1)(\lambda^2 k^2 + 2\lambda k + 4k^2 + 1)\}$. For each $C' \in \mathcal{C}$ and each set $X' \in \mathcal{S}_{C'}$, we use the following steps to update $\mathcal{D}$.

1. Starting from the partition $\mathcal{Q}_0 := \mathcal{R}_{C'}$, we merge all parts that are not in X' together to be one part called O_1. Call the resulting partition $\mathcal{Q}_1$.
2. In the graph $H(\mathcal{Q}_1)$, for each connected component of $H(\mathcal{Q}_1) \setminus \{O_1\}$ that has more than $4k^2 + 1$ vertices, we merge the parts corresponding to the vertices of the component with O_1. Let $\mathcal{Q}_2$ be the resulting partition and let O_2 be the part of $\mathcal{Q}_2$ that contains O_1.
3. In the graph $H(\mathcal{Q}_2)$, for each connected component of $H(\mathcal{Q}_2) \setminus \{O_2\}$, we merge the parts corresponding to vertices in that component. Let $\mathcal{Q}_3$ be the resulting partition.
4. Add the triple $D_{C',X'} := (\mathcal{P}_{\chi(t)}, \mathcal{Q}_{\chi(t)}, O)$ to $\mathcal{D}$, where $\mathcal{P}_{\chi(t)}$, $\mathcal{Q}_{\chi(t)}$ and O are restrictions of $\mathcal{Q}_3$, $\mathcal{Q}_2$, and O_2 to $\chi(t)$, respectively.

This completes the description of our algorithm.

Run-time. We now bound the run time of this algorithm. The family $\mathcal{C}$ can be computed in $\mathsf{O}((4k^2+\mathsf{O}(k^2(\lambda k+1)^5))^{4k^2}n^{\mathsf{O}(1)}) = (\lambda k)^{\mathsf{O}(k^2)}n^{\mathsf{O}(1)}$ time, and the size of $\mathcal{C}$ is $(\lambda k)^{\mathsf{O}(k^2)}\log n$ using Lemma 13. In the next step, Case 1 runs in $n^{\mathsf{O}(1)}$ time for each $C' \in \mathcal{C}$. Case 2 requires $\mathsf{O}((4k^2+1+(4k^2)((\lambda k)^2+2\lambda k+4k^2+1))^{4k^2+1}n^{\mathsf{O}(1)}) = (\lambda k)^{\mathsf{O}(k^2)}n^{\mathsf{O}(1)}$ time to generate $\mathcal{S}_{C'}$ for each $C' \in \mathcal{C}$, and the size of each $\mathcal{S}_{C'}$ is bounded by $(\lambda k)^{\mathsf{O}(k^2)}\log n$. The rest of the steps in Case 2 take $n^{\mathsf{O}(1)}$ time. Therefore, the total time needed for this algorithm is

$$(\lambda k)^{\mathsf{O}(k^2)}n^{\mathsf{O}(1)} + (\lambda k)^{\mathsf{O}(k^2)}\log n \cdot (\lambda k)^{\mathsf{O}(k^2)}n^{\mathsf{O}(1)} = (\lambda k)^{\mathsf{O}(k^2)}n^{\mathsf{O}(1)}.$$

Correctness. We now prove the correctness of the algorithm. We first show that the algorithm indeed outputs a family of nice decompositions.

Claim 1 1. *If $|\chi(t)| \le (\lambda k+1)^5$, then the triple $D_{C'}$ is indeed a nice decomposition for every $C' \in \mathcal{C}$ for which $D_{C'}$ is defined.*
2. *If $|\chi(t)| > (\lambda k+1)^5$, then, for every $C' \in \mathcal{C}$ and $X' \in \mathcal{S}_{C'}$, the triple $D_{C',X'}$ is indeed a nice decomposition.*

Proof Suppose $|\chi(t)| \le (\lambda k+1)^5$ and $\mathcal{R}_{C'}$ has at most $4k^2+1$ parts. Then $\mathcal{R}_{C'}|_{\chi(t)}$ also has at most $4k^2+1$ parts, which verifies condition 2 in Definition 9. The remaining conditions in the definition hold immediately, and hence, $D_{C'}$ is indeed a nice decomposition.

We henceforth consider the case where $|\chi(t)| > (\lambda k+1)^5$. Let $C' \in \mathcal{C}$ be fixed and $X' \in \mathcal{S}_{C'}$, yielding a triple $D_{C',X'} := (\mathcal{P}_{\chi(t)}, \mathcal{Q}_{\chi(t)}, O)$ to be included into $\mathcal{D}$. We will prove that $D_{C',X'}$ satisfies the conditions in Definition 9. We note that $\mathcal{Q}_2$ refines $\mathcal{Q}_3$, so $\mathcal{Q}_{\chi(t)}$ refines $\mathcal{P}_{\chi(t)}$.

For $i = 0, 1, 2$, let H_i' denote the subgraph of $H(\mathcal{Q}_i)$ induced by vertices whose corresponding parts intersect $\chi(t)$. By compactness of the tree decomposition (τ, χ), the subgraph G_t is connected. Therefore, there exists a path in G_t between every pair of vertices of $\chi(t)$. This implies that there exists a path in G_t between every pair of vertices of H_0' as every vertex of H_0' corresponds to a part that intersects $\chi(t)$. Hence, the graph H_0' is connected.

Let us first consider the case where $O = O_2 \cap \chi(t) = \emptyset$. In this case the parts merged to become O_1 do not intersect $\chi(t)$, and hence $H_0' = H_1'$. If $H_0' = H_1'$ has more than $4k^2+1$ vertices, then H_1' will be merged into O_2. Therefore, we conclude that $H_0' = H_1'$ has no more than $4k^2+1$ vertices and $H_0' = H_1' = H_2'$. In step 3, H_2' is in one component of $H(\mathcal{Q}_2)\backslash\{O_2\}$. Therefore, in $\mathcal{Q}_3$, only one part intersects $\chi(t)$ and this part consists of at most $4k^2+1$ parts in $\mathcal{Q}_2$ intersecting $\chi(t)$. This implies $\mathcal{P}_{\chi(t)}$ has only one part, and this part contains at most $4k^2+1$ parts of $\mathcal{Q}_{\chi(t)}$. The rest of the conditions in the definition of nice decomposition hold immediately.

Next we consider the case where $O = O_2 \cap \chi(t) \neq \emptyset$. Since O_2 is a part of $\mathcal{Q}_2$, we know that O is a part of $\mathcal{Q}_{\chi(t)}$, which proves condition (i) of the definition of nice decomposition. Every part of $\mathcal{Q}_3$ consists of at most $4k^2+1$ parts of $\mathcal{Q}_2$ due to step 3, so every part of $\mathcal{P}_{\chi(t)}$ consists of at most $4k^2+1$ parts of $\mathcal{Q}_{\chi(t)}$, proving condition (ii). For two distinct parts P and P' of $\mathcal{Q}_3$, P and P' share either an adhesion or a G_t-edge only if the corresponding vertices in $H(\mathcal{Q}_3)$ are adjacent. In graph $H(\mathcal{Q}_3)$, each edge has one end-vertex being O_2. This implies that one of P and P' has to be O_2. Therefore, no two parts other than O in $\mathcal{P}_{\chi(t)}$ share an adhesion or a G_t-edge, thus proving conditions (iii) and (iv) in the definition of nice decomposition. This completes the Proof of Claim 1. □

The following lemma completes the Proof of Lemma 6.

Lemma 14 *If a k-subpartition Π of $V(G_t)$ witnesses f-correctness of $f_t(\mathcal{P}_{A_t}, \bar{x}, d)$ for some $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, $\bar{x} \in \{0, 1, \ldots, \lambda\}^k$ and $d \in \{0, 1, \ldots, 2k^2\}$, then $\mathcal{D}$ contains a nice decomposition $D = (\mathcal{P}_{\chi(t)}, \mathcal{Q}_{\chi(t)}, O)$ such that $\mathcal{Q}_{\chi(t)}$ refines a restriction of Π to $\chi(t)$.*

Proof We begin with some notations. Let Π be a k-subpartition of $V(G_t)$ that witnesses f-correctness of $f_t(\mathcal{P}_{A_t}, \bar{x}, d)$ for some $\mathcal{P}_{A_t} \in \mathcal{F}^{A_t}$, $\bar{x} \in \{0, 1, \ldots, \lambda\}^k$, and $d \in \{0, 1, \ldots, 2k^2\}$. Then, Π is a restriction of Ω to $V(G_t)$ and $|\delta_T(\Pi)| \leq d \leq 2k^2$. Let $\Pi_{\chi(t)}$ denote a partition of $\chi(t)$ that is a restriction of Π to $\chi(t)$. By definition, the partition $\mathcal{P}_{A_t}$ is a restriction of Π to A_t.

Claim 2 *There exists a partition $\tilde{\Pi}_{\chi(t)}$ of proj$(T, \chi(t))$ such that $\Pi_{\chi(t)}$ is a restriction of $\tilde{\Pi}_{\chi(t)}$ to $\chi(t)$ and $|\delta_{proj(T,\chi(t))}(\tilde{\Pi}_{\chi(t)})| \leq 4k^2$.*

Proof Since $\mathcal{P}_{A_t}$ is T-feasible, there exists a partition $\mathcal{B}$ of V that $2k^2$-respects T and its restriction to A_t is $\mathcal{P}_{A_t}$. Without loss of generality we may assume that $\mathcal{B}$ has exactly k' parts $B_1, \ldots, B_{k'}$ (recall that $\mathcal{P}_{A_t} = (\tilde{P}_1, \ldots, \tilde{P}_{k'})$) such that $B_i \cap A_t = \tilde{P}_i$ for all $i \in [k']$. Moreover, we may assume that Π has exactly k' parts $\pi_1, \ldots, \pi_{k'}$ such that $\pi_i \cap A_t = \tilde{P}_i$ for all $i \in [k']$. With these two partitions, we can define a partition $\mathcal{B}' = (B'_1, \ldots, B'_{k'})$ of V by $B'_i = (B_i \backslash V(G_t)) \cup \pi_i$ for each $i \in [k']$. The partition $\Pi_{\chi(t)}$ is a restriction of $\mathcal{B}'$ to $\chi(t)$. Moreover, there are no edges between $V(G_t) \backslash A_t$ and $V \backslash V(G_t)$, and $\mathcal{P}_{A_t}$ is a restriction of both $\mathcal{B}$ and Π to A_t. Hence, we have $|\delta_T(\mathcal{B}')| \leq |\delta_T(\mathcal{B})| + |\delta_T(\Pi)| \leq 4k^2$.

Now by Lemma 3, given partition $\mathcal{B}'$ of V, there exists a partition $\tilde{\Pi}_{\chi(t)}$ of proj$(T, \chi(t))$ such that its restriction to $\chi(t)$ is $\Pi_{\chi(t)}$ and $|\delta_{\text{proj}(T,\chi(t))}(\tilde{\Pi}_{\chi(t)})| \leq |\delta_T(\mathcal{B}')| \leq 4k^2$. This completes the Proof of Claim 2. □

From now on we will fix $\tilde{\Pi}_{\chi(t)}$ to be a partition of $V(\text{proj}(T, \chi(t)))$ that satisfies the conditions of Claim 2. Let $\tilde{C}$ denote the set of edges $\delta_{\text{proj}(T,\chi(t))}(\tilde{\Pi}_{\chi(t)})$, which is a subset of $E(\text{proj}(T, \chi(t)))$. It follows that $|\tilde{C}| \leq 4k^2$.

Removing $\tilde{C}$ from proj$(T, \chi(t))$ yields a partition of $V(\text{proj}(T, \chi(t)))$ whose parts are connected components of proj$(T, \chi(t)) \backslash C$. We denote this partition as $\tilde{\Pi}'_{\chi(t)}$, and observe that $\tilde{\Pi}'_{\chi(t)}$ is necessarily a refinement of $\tilde{\Pi}_{\chi(t)}$. Moreover, we may assume that each part of $\tilde{\Pi}'_{\chi(t)}$ intersects $\chi(t)$. If some parts of $\tilde{\Pi}'_{\chi(t)}$ do not intersect $\chi(t)$, then there exist two parts P and P' of $\tilde{\Pi}'_{\chi(t)}$ such that P intersects $\chi(t)$ while P' does not, and there is an edge in $\tilde{C} \subseteq E(\text{proj}(T, \chi(t)))$ with one end-vertex in P and the other end-vertex in P'. We can modify $\tilde{\Pi}_{\chi(t)}$ so that P' belongs to the part containing

Table 1 Notations used for the Proof of Lemma 14

Π	k-subpartition of $V(G_t)$
$\Pi_{\chi(t)}$	Partition of $\chi(t)$
$\tilde{\Pi}_{\chi(t)}$	Partition of $V(\text{proj}(T, \chi(t)))$
$\tilde{\Pi}'_{\chi(t)}$	Partition of $V(\text{proj}(T, \chi(t)))$
$\tilde{C}$	Subset of $E(\text{proj}(T, \chi(t)))$
$S_{\chi(t)}$	Subset of $\chi(t)$
$\tilde{S}$	Subset of $V(\text{proj}(T, \chi(t)))$
$\tilde{S}_E$	Subset of $E(\text{proj}(T, \chi(t)))$

P. After such modification, the size of $\delta_{\text{proj}(T,\chi(t))}(\tilde{\Pi}_{\chi(t)})$ does not increase (because grouping P' into the part containing P does not require us to cut any edge not in $\tilde{C}$) and $\Pi_{\chi(t)}$ is still a restriction of $\tilde{\Pi}_{\chi(t)}$ to $\chi(t)$. By doing this repeatedly, we may assume that each part of $\tilde{\Pi}'_{\chi(t)}$ intersects $\chi(t)$ while the size of $\tilde{C}$ does not increase.

Since Ω is an optimum MINMAX k-CUT in G, we have that $|\delta_G(\Omega)| \le k\text{OPT} \le k\lambda$. Since $\Pi_{\chi(t)}$ is a restriction of Ω to $\chi(t)$, by the edge-unbreakability property of $\chi(t)$, we know that at most one part of $\Pi_{\chi(t)}$ has size exceeding $(\lambda k + 1)^5$. Now, let $S_{\chi(t)}$ denote the union of the parts of $\Pi_{\chi(t)}$ whose sizes are at most $(\lambda k + 1)^5$, i.e.,

$$S_{\chi(t)} := \bigcup_{\substack{\pi \text{ is a part of } \Pi_{\chi(t)},\\ |\pi| \le (\lambda k+1)^5}} \pi.$$

Furthermore, we use $\tilde{S}$ to denote the union of parts of $\tilde{\Pi}_{\chi(t)}$ whose intersection with $S_{\chi(t)}$ is non-empty. Each part of $\tilde{\Pi}_{\chi(t)}$ included in $\tilde{S}$ induces a set of edges in $\text{proj}(T, \chi(t))$ whose both end-vertices are in this part. We use $\tilde{S}_E$ to denote the union of such edges, i.e.,

$$\tilde{S} := \bigcup_{\substack{\pi \text{ is a part of } \tilde{\Pi}_{\chi(t)},\\ \pi \cap S_{\chi(t)} \ne \emptyset}} \pi \text{ and}$$
$$\tilde{S}_E := \bigcup_{\substack{\pi \text{ is a part of } \tilde{\Pi}_{\chi(t)},\\ \pi \cap S_{\chi(t)} \ne \emptyset}} E(\text{proj}(T, \chi(t))[\pi]).$$

We remark that $\tilde{S}_E \cap \tilde{C} = \emptyset$ because every edge in $\tilde{C}$ has end-vertices in different parts of $\tilde{\Pi}_{\chi(t)}$. For convenience, we summarize the nature of notations introduced here in Table 1.

Now that we have introduced these notations and definitions, our next goal is to bound the size of $\tilde{S}_E$. We will use the following claim.

Claim 3 *For every part P of $\tilde{\Pi}'_{\chi(t)}$, we have that $|P| \le 2(|P \cap \chi(t)| + 4k^2 + 1)$.*

Proof If P is a part of $\tilde{\Pi}'_{\chi(t)}$, then by definition of $\tilde{\Pi}'_{\chi(t)}$, the subgraph induced by P in $\mathrm{proj}(T, \chi(t))$, i.e., $\mathrm{proj}(T, \chi(t))[P]$, is a connected subtree of $\mathrm{proj}(T, \chi(t))$. To bound the size of P, we will bound the sizes of the following types of vertices:

1. vertices of P with degree at least 3 in $\mathrm{proj}(T, \chi(t))[P]$,
2. vertices of P that are not in $\chi(t)$ and have degree at most 2 in the subtree $\mathrm{proj}(T, \chi(t))[P]$, and
3. vertices in $P \cap \chi(t)$.

These three types together form a superset of the vertices of P.

In order to bound the number of Type 2 vertices, we note that these vertices are in $V(\mathrm{proj}(T, \chi(t)))\backslash\chi(t)$, which means they are of degree at least 3 in $\mathrm{proj}(T, \chi(t))$. For each Type 2 vertex, some edge in $\mathrm{proj}(T, \chi(t))$ adjacent to it which connects P to some other component is not included in $\mathrm{proj}(T, \chi(t))[P]$, resulting it to have degree at most 2 in $\mathrm{proj}(T, \chi(t))[P]$. Since each part of $\tilde{\Pi}'_{\chi(t)}$ induces a connected subtree of $\mathrm{proj}(T, \chi(t))$, each Type 2 vertex serves to connect P to a unique part of $\tilde{\Pi}'_{\chi(t)}$. Here $\tilde{\Pi}'_{\chi(t)}$ has at most $4k^2 + 1$ parts as $|\tilde{C}| \leq 4k^2$, and hence the number of Type 2 vertices is at most $4k^2 + 1$.

In order to bound the number of Type 1 vertices, we will first bound the number of leaves of $\mathrm{proj}(T, \chi(t))[P]$. Leaves of $\mathrm{proj}(T, \chi(t))[P]$ are either in $\chi(t)$ or not in $\chi(t)$. The number of leaves in $\chi(t)$ is at most $|P \cap \chi(t)|$. The leaves of $\mathrm{proj}(T, \chi(t))[P]$ that are not in $\chi(t)$ are Type 2 vertices. So the total number of leaves of $\mathrm{proj}(T, \chi(t))[P]$ is at most $|P \cap \chi(t)| + 4k^2 + 1$.

Next we bound the number of Type 1 vertices. We have the following inequality, where all degrees are with respect to the subgraph $\mathrm{proj}(T, \chi(t))[P]$:

$$\begin{aligned}
2|P| - 2 = 2|E(\mathrm{proj}(T, \chi(t))[P])| &= \sum_{v \in P} \deg(v) \\
&\geq |\{v \in P : \deg(v) = 1\}| + 2|\{v \in P : \deg(v) = 2\}| \\
&\quad + 3|\{v \in P : \deg(v) \geq 3\}| \\
&= 2|P| - |\{v \in P : \deg(v) = 1\}| + |\{v \in P : \deg(v) \geq 3\}|.
\end{aligned}$$

where the last equation holds due to the fact that $|\{v \in P : \deg(v) = 1\}| + |\{v \in P : \deg(v) = 2\}| + |\{v \in P : \deg(v) \geq 3\}| = |P|$. This implies that the number of Type 1 vertices can be bounded by the following relationship:

$$|\{v \in P : \deg(v) \geq 3\}| \leq |\{v \in P : \deg(v) = 1\}| \leq |P \cap \chi(t)| + 4k^2 + 1.$$

The number of Type 3 vertices is exactly $|P \cap \chi(t)|$, and hence the size of P is at most

$$(4k^2 + 1) + (|P \cap \chi(t)| + 4k^2 + 1) + |P \cap \chi(t)| = 2(|P \cap \chi(t)| + 4k^2 + 1).$$

□

Claim 4 $|\tilde{S}|, |\tilde{S}_E| \leq (4k^2 + 1) \cdot 2((\lambda k + 1)^5 + 4k^2 + 1).$

Proof We will start by bounding the size of $\tilde{S}$. Let us fix π to be a part of $\tilde{\Pi}_{\chi(t)}$ such that $\pi \cap S_{\chi(t)} \neq \emptyset$, and π' to be a part of $\tilde{\Pi}'_{\chi(t)}$ contained in π. Here, we notice that $\pi \cap \chi(t)$ is a part of $\Pi_{\chi(t)}$ and $|\pi \cap \chi(t)| \leq (\lambda k + 1)^5$. Then by Claim 3, we know that

$$\begin{aligned}|\pi'| &\leq 2(|\pi' \cap \chi(t)| + 4k^2 + 1)\\ &\leq 2(|\pi \cap \chi(t)| + 4k^2 + 1) \leq 2((\lambda k + 1)^5 + 4k^2 + 1).\end{aligned}$$

The set $\tilde{S}$ is the union of all such parts π', i.e., it is the union of parts π' of $\tilde{\Pi}'_{\chi(t)}$ where π' is contained in some part π of $\tilde{\Pi}_{\chi(t)}$ such that $\pi \cap S_{\chi(t)} \neq \emptyset$. There are at most $4k^2 + 1$ such candidates for π' because $\tilde{\Pi}'_{\chi(t)}$ has at most $4k^2 + 1$ parts. Hence,

$$|\tilde{S}| \leq (4k^2 + 1) \cdot 2((\lambda k + 1)^5 + 4k^2 + 1).$$

To bound the size of $\tilde{S}_E$, we observe that $\tilde{S}_E$ forms a forest over the vertex set $\tilde{S}$, and thus $|\tilde{S}_E| \leq |\tilde{S}| \leq (4k^2 + 1) \cdot 2((\lambda k + 1)^5 + 4k^2 + 1)$. □

The first step of our algorithm generates a family $\mathcal{C}$ using Lemma 13 on the set $E(\text{proj}(T, \chi(t)))$ with $s_1 = \min\{|E(\text{proj}(T, \chi(t)))|, 4k^2\}$ and $s_2 = \min\{|E(\text{proj}(T, \chi(t)))|, (4k^2+1)\cdot 2((\lambda k+1)^5+4k^2+1)\}$. By Claim 4 and Lemma 13, this implies that $\mathcal{C}$ contains a set C such that $\tilde{C} \subseteq C$ and $C \cap \tilde{S}_E = \emptyset$. For the rest of the proof, let us fix $C \in \mathcal{C}$ such that $\tilde{C} \subseteq C$ and $C \cap \tilde{S}_E = \emptyset$. We now introduce some more more notations and prove certain useful properties of C.

Here, we note that the partition $\mathcal{R}_C$ refines $\tilde{\Pi}'_{\chi(t)}$ because $\tilde{C} \subseteq C$. We use L_C to denote the set of parts of $\mathcal{R}_C$ that are contained in $\tilde{S}$. We use N_C to denote the set of parts of $\mathcal{R}_C$ outside $\tilde{S}$ that either share adhesion with some part in L_C or share G_t-edges with some part in L_C. We will use the following observation and claim to bound the size of N_C.

Observation 3 *Every edge in $C \backslash \tilde{C}$ necessarily has both end-vertices in $V(proj(T, \chi(t))) \backslash \tilde{S}$. This is because every edge between $V(proj(T, \chi(t))) \backslash \tilde{S}$ and $\tilde{S}$ belongs to $\tilde{C}$, and every edge whose both end-vertices are in $\tilde{S}$ are either in $\tilde{S}_E$ (which does not intersect C) or in $\tilde{C}$. This implies that when restricted to $\tilde{S}$, the partition $\mathcal{R}_C$ and $\tilde{\Pi}'_{\chi(t)}$ are the same.*

Claim 5 $|N_C| \leq (4k^2 + 1)((\lambda k)^2 + 2\lambda k + 4k^2 + 1)$.

Proof Let us fix one part R in L_C and bound the number of parts of $\mathcal{R}_C$ that could share adhesion or G_t-edge with R. By Observation 3, we know that R is also a part of $\tilde{\Pi}'_{\chi(t)}$. We will use π to denote the part of $\tilde{\Pi}_{\chi(t)}$ that contains R. The parts of $\mathcal{R}_C$ that share either an adhesion or a G_t-edge with R can be enumerated by the following four types:

1. parts outside π that share adhesion with R via $A_{t'}$, where t' is a child of t,
2. parts outside π that share adhesion with R via A_t,
3. parts outside π that share G_t-edge with R, and

4. parts in π.

We bound the number of parts of Type 1. For this, we will first bound the number of children t' of t such that $A_{t'}$ intersects both R and some part outside π. Let t' be a child of t such that $A_{t'}$ intersects R and a part R' that is outside π. Then $R \cap A_{t'}$ and $R' \cap A_{t'}$ are in different parts of Π. By compactness of τ, we know that $R \cap A_{t'}$ has a neighbor v in $V(G_{t'}) \backslash A_{t'}$, and $R' \cap A_{t'}$ has a neighbor v' in $V(G_{t'}) \backslash A_{t'}$. Since $V(G_{t'}) \backslash A_{t'}$ induces a connected subgraph in G, there is a path between v and v' in $G[V(G_{t'}) \backslash A_{t'}]$. Hence, in order to separate $R \cap A_{t'}$ and $R' \cap A_{t'}$, the k-subpartition Π must cut some edge with at least one end-vertex in $V(G_{t'}) \backslash A_{t'}$. We fix one such edge and denote it as $e_{t'}$. Then $e_{t'}$ is contained in $\delta_G(\Pi)$. We associate one such edge $e_{t'}$ for each child t' of t such that $A_{t'}$ intersects R and a part R' that is outside π.

We now show that the edge $e_{t'}$ associated the child t' is unique. For the sake of contradiction, suppose that $e_{t'} = e_{t''} = e$ for two children t' and t'' of t. Let $e = \{u', u''\}$ with $u' \in V(G_{t'}) \setminus A_{t'}$ and $u'' \in V(G_{t''}) \setminus A_{t''}$. Then, e is contained in some bag $\chi(t_0)$. The bags containing u' induce a connected subtree, and $u' \notin \chi(t)$, so t_0 must be a descendant of t' (inclusive). Similarly t_0 must be a descendant of t'' (inclusive). This is a contradiction because t' and t'' are distinct children of t.

Therefore, the number of children t' of t such that $A_{t'}$ intersects both R and some other part outside π is at most $|\delta_G(\Pi)| \leq \lambda k$. Each such adhesion has size at most λk, so it contributes at most λk to the number of adjacent parts that R could have. Hence the size of type 1 is bounded above by $(\lambda k)^2$.

In order to bound the number of parts of Type 2, we use the fact that $|A_t| \leq \lambda k$ and conclude that the the number of parts of Type 2 is at most λk.

In order to bound the number of parts of Type 3, we observe that a G_t-edge with one end-vertex in π and the other end-vertex not in π is necessarily in $\delta_G(\Pi)$. Each part outside π that shares G_t-edge with R connects R to a unique edge in $\delta_G(\Pi)$, and hence the number of parts of Type 3 is at most $|\delta_G(\Pi)| \leq \lambda k$.

Lastly, we bound the number of parts of Type 4. Since R is a part in L_C, by definition we know that π is contained in $\tilde{S}$. This means that every part in π is also a part of $\tilde{\Pi}'_{\chi(t)}$ by Observation 3. Therefore, the size of type 4 is at most $4k^2 + 1$.

We conclude that R shares an adhesion or a G_t-edge with at most $(\lambda k)^2 + \lambda k + \lambda k + 4k^2 + 1$ parts of $\mathcal{R}_C$. Hence, the size of N_C is at most

$$|L_C| \cdot ((\lambda k)^2 + \lambda k + \lambda k + (4k^2 + 1)).$$

Since parts in L_C are also parts in $\tilde{\Pi}'_{\chi(t)}$, we know that $|L_C| \leq 4k^2 + 1$. This yields the desired bound:

$$|N_C| \leq (4k^2 + 1)((\lambda k)^2 + 2\lambda k + 4k^2 + 1).$$

□

We now have the ingredients to show that $\mathcal{D}$ contains a nice decomposition $D = (\mathcal{P}_{\chi(t)}, \mathcal{Q}_{\chi(t)}, O)$ such that $\mathcal{Q}_{\chi(t)}$ refines $\Pi_{\chi(t)}$. We begin with the easier case where the size of the bag is small.

Claim 6 *If $|\chi(t)| \le (\lambda k+1)^5$, then the nice decomposition $D_C = ((\chi(t)), \mathcal{R}_C|_{\chi(t)}, \emptyset)$ is contained in $\mathcal{D}$ and the partition $\mathcal{R}_C|_{\chi(t)}$ refines $\Pi_{\chi(t)}$.*

Proof If $|\chi(t)| \le (\lambda k + 1)^5$, then we note that $S_{\chi(t)} = \chi(t)$ and hence $\tilde{S} = V(\mathrm{proj}(T, \chi(t)))$. By Observation 3, we know that $\mathcal{R}_C = \tilde{\Pi}'_{\chi(t)}$. This guarantees that $\mathcal{R}_C$ has at most $4k^2 + 1$ parts, and thus the triple $((\chi(t)), \mathcal{R}_C|_{\chi(t)}, \emptyset)$ is defined and added into $\mathcal{D}$. Moreover, we know that $\mathcal{R}_C = \tilde{\Pi}'_{\chi(t)}$ refines $\tilde{\Pi}_{\chi(t)}$, and hence $\mathcal{R}_C|_{\chi(t)}$ refines $\Pi_{\chi(t)}$. □

We now handle the case where the size of the bag is large. For the rest of the proof, suppose that $|\chi(t)| > (\lambda k + 1)^5$. We recall that $\tilde{C} \subseteq C$ and $C \cap \tilde{S}_E = \emptyset$. Since $|L_C| \le 4k^2 + 1$, by Claim 5 and Lemma 13, the family $\mathcal{S}_C$ is guaranteed to contain a set X such that $L_C \subseteq X$ and $X \cap N_C = \emptyset$. Let us fix such a set X in the remainder of the proof. The next claim states that the nice decomposition $D_{C,X}$ yields a refinement of $\Pi_{\chi(t)}$ as desired, thereby completing the Proof of Lemma 14. □

Claim 7 *If $|\chi(t)| > (\lambda k+1)^5$, then the nice decomposition $D_{C,X} = (\mathcal{P}_{\chi(t)}, \mathcal{Q}_{\chi(t)}, O)$ yields a partition $\mathcal{Q}_{\chi(t)}$ of $\chi(t)$ that refines $\Pi_{\chi(t)}$.*

Proof By definition of the set X, at the end of step 1 of the algorithm, the part O_1 contains all parts in N_C and no parts in L_C. Moreover, in the graph $H(\mathcal{Q}_1)$, we have $N_{H(\mathcal{Q}_1)}(L_C) = \{O_1\}$. Since L_C contains at most $4k^2 + 1$ parts, we know that no part in L_C is merged into O_1 in step 2. Therefore, every part of $\mathcal{R}_C$ in L_C remains a single part in $\mathcal{Q}_2$. This implies that every part of $\tilde{\Pi}'_{\chi(t)}$ in $\tilde{S}$ remains a single part in $\mathcal{Q}_2$. If a part π of $\tilde{\Pi}_{\chi(t)}$ is contained in $\tilde{S}$, then π is the union of some parts of $\tilde{\Pi}'_{\chi(t)}$ in $\tilde{S}$, and hence the union of some parts of $\mathcal{Q}_2$. If a part π of $\tilde{\Pi}_{\chi(t)}$ is not contained in $\tilde{S}$, then π is the unique part of $\tilde{\Pi}_{\chi(t)}$ that is not contained in $\tilde{S}$. This implies that π is the union of parts of $\mathcal{Q}_2$ that are not contained in L_C. Thus, we conclude that $\mathcal{Q}_2$ refines $\tilde{\Pi}_{\chi(t)}$, and hence $\mathcal{Q}_{\chi(t)}$ refines $\Pi_{\chi(t)}$. □

4 Reduction to unweighted instances with logarithmic optimum value

In this section, we show a $(1+\epsilon)$-approximation preserving reduction to unweighted instances with optimum value $O((k/\epsilon^3)\log n)$. The ideas in this section are somewhat standard and are also the building blocks for the $(1+\epsilon)$-approximation for Minsum k-cut. Our contribution to the reduction is simply showing that the ideas also apply to Minmax k-cut.

Theorem 5 *There exists an algorithm that takes as input a weighted instance $G = (V, E)$ of* Minmax k-cut, *namely an n-vertex simple graph $G = (V, E)$ with edge weights $w : E \to \mathbb{Z}_+$, an integer $k \ge 2$, and an $\epsilon \in (0, 1)$, and runs in time poly$(n, 1/\epsilon)\log^2\left(\sum_{e\in E} w_e\right)$ to return a collection $\mathcal{C}$ of unweighted instances of* Minmax k-cut *such that with high probability*

(i) *for each instance in $\mathcal{C}$, the number of vertices is at most n and the number of edges is $O(n^4/\epsilon)$,*

(ii) *the number of instances in* $\mathcal{C}$ *is* $O(\log(\sum_{e\in E} w_e)\log(n/\epsilon))$, *and*
(iii) *there exists an instance* $H \in \mathcal{C}$ *satisfying (a)* $OPT(H,k) = O((k/\epsilon^3)\log n)$ *and (b) every optimum* k*-partition* $\mathcal{P}_H$ *in* H *can be used to recover a* k*-partition* $\mathcal{P}$ *of the vertices of* G *such that* $cost_{G_w}(\mathcal{P}) = (1+O(\epsilon))OPT(G_w,k)$ *in time* $n^{O(1)}$.

Proof We use Algorithm 3 to prove Theorem 5. Let the input instance be $G=(V,E)$ with edge weights $w: E \to \mathbb{Z}_+$ along with an integer $k \geq 2$ and $\epsilon \in (0,1)$.

Algorithm 3 Algorithm for the Proof of Theorem 5

```
$\mathcal{C} \leftarrow \emptyset$
for $\lambda = 1, 2, 2^2, \ldots, 2^{\log(\sum_{e\in E} w_e)}$ do
    Compute an unweighted multigraph $H = (V', E')$ using Lemma 15 with value $\lambda$
    for $\lambda' = 1, 2, 2^2, \ldots, 2^{\log|E'|}$ do
        count $\leftarrow 0$
        while count $< k-1$ and $H$ has a cut set $F \subseteq E'$ such that $0 < |F| \leq \epsilon\lambda'/(2(k-1))$ do
            $H \leftarrow H - F$, count $\leftarrow$ count $+ 1$
        end while
        if count $= k-1$ then
            Go to the next choice of $\lambda'$
        end if
        $H_1 \leftarrow H$
        Construct $H_2$ from $H_1$ by sampling each edge with probability $p = 1000\log|V'|/\epsilon^2\mu$, where
        $\mu := \min\{OPT(Q,2) : Q$ is a component of $H_1\}$
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{H_2\}$
    end for
end for
Return $\mathcal{C}$
```

We bound the run time of the algorithm. For each fixed $\lambda \in \{1,2,2^2,\ldots,2^{\log(\sum_{e\in E} w_e)}\}$, the unweighted multigraph $H=(V',E')$ can be constructed in poly$(n,1/\epsilon)\sum_{e\in E}\log w(e)$ time, and satisfies $|E'| \leq 8|E|^2/\epsilon$ by Lemma 15. Each inner for-loop can be implemented to run in poly$(n,1/\epsilon)$ time. Therefore, the total run time is at most

$$\log^2\left(\sum_{e\in E} w_e\right)\text{poly}(n,1/\epsilon)\cdot\log(8|E|^2/\epsilon) = \text{poly}(n,1/\epsilon)\log^2\left(\sum_{e\in E} w_e\right).$$

Next, we prove the correctness of Algorithm 3. We need to show that collection $\mathcal{C}$ of unweighted instances constructed by the algorithm satisfies properties (i), (ii), and (iii). For each $\lambda \in \{1,2,2^2,\ldots,2^{\log(\sum_{e\in E} w_e)}\}$, the unweighted multigraph $H=(V',E')$ has at most $8|E|^2/\epsilon \leq 8n^4/\epsilon$ edges by Lemma 15. Each instance added to $\mathcal{C}$ in this outer for-loop is a subgraph of H, and thus has at most $8n^4/\epsilon$ edges. This proves property (i). The total number of inner for-loops is $\log(\sum_{e\in E} w_e)\log(8n^4/\epsilon)$, and at most one instance is added to $\mathcal{C}$ in each inner for-loop. Thus, we have

$$|\mathcal{C}| = O\left(\log\left(\sum_{e\in E} w_e\right)\log(n/\epsilon)\right),$$

and this implies property (ii).

We now prove property (iii). We note that there exists $\lambda \in \{1,2,2^2,\ldots,2^{\log(\sum_{e\in E} w_e)}\}$ such that $\lambda \in [\text{OPT}(G_w,k), 2\text{OPT}(G_w,k)]$. Let us fix this choice of

λ henceforth. Let $H = (V', E')$ be the unweighted multigraph constructed using Lemma 15 with value λ. We also fix the choice of $\lambda' \in \{1, 2, 2^2, \ldots, 2^{\log(|E'|)}\}$ such that $\lambda' \in [\mathrm{OPT}(H, k), 2\mathrm{OPT}(H, k)]$.

For this choice of λ', the algorithm repeatedly removes cut sets F (i.e., a subset F of edges that cross some 2-partition of G) that have $0 < |F| \leq \epsilon\lambda'/(2(k-1))$. If the algorithm can remove $k-1$ such cut sets, then it would have removed at most $\epsilon\lambda'/2 \leq \epsilon\mathrm{OPT}(H, k)$ edges while creating k connected components, thus contradicting the optimum value. Hence, the while loop of the algorithm will terminate with count $< k-1$ and with the number of edges being removed in the while loop being less than $\epsilon\lambda'/2 \leq \epsilon\mathrm{OPT}(H, k)$. Hence, we will have a subgraph $H_1 = (V', E_1)$ of H such that (1) $|E' - E_1| \leq \epsilon\mathrm{OPT}(H, k)$ and (2) the min-cut in each connected component of H_1 is at least $\epsilon\mathrm{OPT}(H, k)/k \geq \epsilon\mathrm{OPT}(H_1, k)/k$. We note that if we can find a $(1 + \mathsf{O}(\epsilon))$-approximate optimum minmax k-partition $(S_1, \ldots, S_k)$ for the unweighted instance H_1, then $\mathrm{cost}_H(S_1, \ldots, S_k) \leq \mathrm{cost}_{H_1}(S_1, \ldots, S_k) + |E' - E_1| \leq \mathrm{cost}_{H_1}(S_1, \ldots, S_k) + \epsilon\mathrm{OPT}(H, k) \leq (1 + \mathsf{O}(\epsilon))\mathrm{OPT}(H, k)$. We note that the components of H_1 are *well-connected*: each component of H_1 has min-cut value at least $\epsilon\mathrm{OPT}(H_1, k)/k$.

The final step of the algorithm constructs H_2 from H_1 by subsampling the edges of H_1: it picks each edge with probability $p = 1000\log|V'|/(\epsilon^2\mu)$, where n is the number of vertices in H_1 and $\mu := \min\{OPT(Q, 2) : Q \text{ is a component of } H_1\}$. Let $H_2 := (V', E_2)$.

By Karger [18], we know that for each component Q of H_1, with probability at least $1 - 1/|V(Q)|^{p/p_Q}$, we have $|\delta_{H_2}(S)|/p \in [(1-\epsilon)|\delta_{H_1}(S)|, (1+\epsilon)|\delta_{H_1}(S)|]$ for every subset $S \subseteq V(Q)$, where $p_Q := 100\log|V(Q)|/(\epsilon^2\mathrm{OPT}(Q, 2))$. Let $Q_1, \ldots, Q_t$ be the components of H_1 with at least 2 vertices. Then, by union bound, with probability at least

$$1 - \sum_{i=1}^{t} \frac{1}{|V(Q_i)|^{p/p_{Q_i}}}$$

we have that $|\delta_{H_2}(S)|/p \in [(1-\epsilon)|\delta_{H_1}(S)|, (1+\epsilon)|\delta_{H_1}(S)|]$ for every subset $S \subseteq V'$. We note that the components with a single vertex in H_1 do not contain any edges, and thus do not affect our argument. We bound this probability as follows:

$$\begin{aligned}
\sum_{i=1}^{t} \frac{1}{|V(Q_i)|^{p/p_{Q_i}}} &= \sum_{i=1}^{t} \frac{1}{|V(Q_i)|^{10\frac{\log|V'|}{\log|V(Q_i)|}\cdot\frac{\mathrm{OPT}(Q_i,2)}{\mu}}} \\
&= \sum_{i=1}^{t} \frac{1}{2^{10\log n\cdot\frac{\mathrm{OPT}(Q_i,2)}{\mu}}} \\
&\leq \sum_{i=1}^{t} \frac{1}{2^{10\log|V'|}} \qquad \text{(by definition of } \mu) \\
&= \sum_{i=1}^{t} \frac{1}{|V'|^{10}} \\
&\leq \frac{1}{|V'|^{9}}.
\end{aligned}$$

Moreover, for each $i \in [t]$, the number of edges in $E(Q_i)$ that survive into H_2 is $\mathsf{O}(\log|V'|/\epsilon^2)|E(Q_2)|$ with probability at least $1-\mathsf{O}(\frac{1}{|V'|^{10}})$ by Chernoff bound. Thus, applying union bound again, with probability at least $1-\mathsf{O}(\frac{1}{|V'|^9}) \geq 1-\mathsf{O}(\frac{1}{n^9})$, (I) the scaled cut-value of every 2-partition is preserved within a $(1\pm\epsilon)$-factor, i.e., $|\delta_{H_2}(S)|/p \in [(1-\epsilon)|\delta_{H_1}(S)|, (1+\epsilon)|\delta_{H_1}(S)|]$ for every $S \subseteq V$ and (II) $|E_2| = \mathsf{O}(\log|V'|/\epsilon^2)|E_1|$.

We show that the instance H_2 in the collection $\mathcal{C}$ satisfies property (iii). The preservation of cut values immediately implies that $\mathrm{OPT}(H_2,k)/p \in [(1-\epsilon)\mathrm{OPT}(H_1,k), (1+\epsilon)\mathrm{OPT}(H_1,k)]$ and moreover,

$$\begin{aligned}
\mathrm{OPT}(H_2,k) &\leq p(1+\epsilon)\mathrm{OPT}(H_1,k)\\
&= \mathsf{O}(1)\frac{\log|V'|}{\epsilon^2\mu}\cdot \mathrm{OPT}(H_1,k)\\
&= \mathsf{O}(1)\frac{k\log|V'|}{\epsilon^3} \qquad \left(\text{since } \mu \geq \frac{\epsilon\mathrm{OPT}(H_1,k)}{k}\right)\\
&\leq \mathsf{O}(1)\frac{k\log n}{\epsilon^3}.
\end{aligned}$$

Thus, we obtain an instance (H_2,k) whose optimum cost is $\mathsf{O}((k/\epsilon^3)\log n)$. Suppose that $(S_1,\ldots,S_k)$ is an optimum k-partition for the instance (H_2,k). Then, by preservation of cut values, the partition $(S_1,\ldots,S_k)$ is a $(1+\mathsf{O}(\epsilon))$-approximate optimum minmax k-partition on the instance (H_1,k). Consequently, $(S_1,\ldots,S_k)$ is a $(1+\mathsf{O}(\epsilon))$-approximate optimum minmax k-partition on the unweighted instance H because

$$\begin{aligned}
\mathrm{cost}_H(S_1,\ldots,S_k) &\leq \mathrm{cost}_{H_1}(S_1,\ldots,S_k) + |E'-E_1|\\
&\leq \mathrm{cost}_{H_1}(S_1,\ldots,S_k) + \epsilon\mathrm{OPT}(H,k)\\
&\leq (1+\mathsf{O}(\epsilon))\mathrm{OPT}(H_1,k) + \epsilon\mathrm{OPT}(H,k)\\
&\leq (1+\mathsf{O}(\epsilon))\mathrm{OPT}(H,k).
\end{aligned}$$

According to Lemma 15, this allows us to recover a $(1+\mathsf{O}(\epsilon))$-approximate minmax k-partition in G_w in time $n^{\mathsf{O}(1)}$, thus proving property (iii). ◻

We emphasize that a constant factor approximation algorithm for Minmax k-cut would help in shaving off a $\log(\sum_{e\in E} w_e)$ term from the run-time given in Theorem 5.

For the sake of completeness, we now give the details of the knapsack PTAS-style rounding procedure to reduce the problem in a $(1+\epsilon)$-approximation preserving fashion to an unweighted instance.

Lemma 15 *There exists an algorithm that takes as input a graph $G=(V,E)$ with edge weights $w: E \to \mathbb{Z}_+$, an $\epsilon \in (0,1)$, and a value λ, and and runs in time poly$(n, 1/\epsilon)\log\left(\sum_{e\in E} w(e)\right)$ to return an unweighted multigraph $H=(V',E')$ such that $|V'| \leq |V|$ and $|E'| \leq 8|E|^2/\epsilon$. Moreover, if $\lambda \in [OPT(G_w,k), 2OPT(G_w,k)]$, then an α-approximate minmax k-partition in H can be used to recover an $\alpha(1+\epsilon)$-approximate minmax k-partition in G_w in time $n^{\mathsf{O}(1)}$ for any $\alpha \geq 1$.*

Proof We contract all edges $e \in E$ with $w(e) > \lambda$ and denote the resulting graph as $G^0 = (V^0, E^0)$. Let $m := |E^0|$, $\theta = \epsilon\lambda/4m$ and $w'(e) := \lceil w(e)/\theta \rceil$ for every $e \in E^0$. We construct the graph $H = (V^0, E')$ by creating $w'(e)$ copies of each edge $e \in E^0$. The time to construct H is $\text{poly}(n, 1/\epsilon) \cdot \log\left(\sum_{e\in E} w(e)\right)$. The bound on the number of vertices in H is due to contraction. We bound the number of edges in H as follows:

$$\begin{aligned}|E'| &= \sum_{e\in E^0} \left\lceil \frac{w(e)}{\theta} \right\rceil \le \sum_{e\in E^0} \left(\frac{w(e)}{\theta} + 1\right) = \frac{w(E^0)}{\theta} + m \\ &= \frac{4w(E^0)m}{\epsilon\lambda} + m \le \frac{4m^2}{\epsilon} + m \le \frac{8m^2}{\epsilon}.\end{aligned}$$

The last but one inequality above is because $w(e) \le \lambda$ for every $e \in E^0$. The bound on $|E'|$ follows now by observing that $m \le |E|$.

Next, suppose that $\lambda \in [\text{OPT}(G_w, k), 2\text{OPT}(G_w, k)]$. We will show that an α-approximate minmax k-partition in H can be used to recover an $\alpha(1+\epsilon)$-approximate minmax k-partition in G_w in polynomial time for any $\alpha \ge 1$. We may assume that $w(e) > \lambda$ for all edges $e \in E$ since all edges $e \in E$ with $w(e) > \lambda$ do not cross an optimum minmax k-partition. Now, we show that $\theta\text{OPT}(H, k) \le (1+\epsilon)\text{OPT}(G_w, k)$. Let $(S_1^*, \ldots, S_k^*)$ be an optimum minmax k-partition in G_w. Then, for every $i \in [k]$, we have that

$$\begin{aligned}\theta w'(\delta(S_i^*)) &= \sum_{e\in\delta(S_i^*)} \left\lceil \frac{w(e)}{\theta} \right\rceil \theta \le \sum_{e\in\delta(S_i^*)} \left(\frac{w(e)}{\theta} + 1\right)\theta = w(\delta(S_i^*)) + \theta|\delta(S_i^*)| \\ &\le \text{OPT}(G_w, k) + \frac{\epsilon\lambda|\delta(S_i^*)|}{4m} \le \text{OPT}(G_w, k)(1+\epsilon).\end{aligned}$$

Hence, $(S_1^*, \ldots, S_k^*)$ is a k-partition of the vertices of H with $\theta\text{cost}_H(S_1^*, \ldots, S_k^*) \le \text{OPT}(G_w, k)(1+\epsilon)$.

Let $(S_1, \ldots, S_k)$ be an α-approximate minmax k-partition in H. For every $i \in [k]$, we have

$$\begin{aligned}w(\delta(S_i)) &= \sum_{e\in\delta(S_i)} w(e) = \sum_{e\in\delta(S_i)} \left(\frac{w(e)}{\theta w'(e)}\right)\theta w'(e) \le \sum_{e\in\delta(S_i)} \theta w'(e) \\ &\le \alpha\text{OPT}(H, k)\theta \le \alpha(1+\epsilon)\text{OPT}(G_w, k).\end{aligned}$$

Thus, $\text{cost}_{G_w}(S_1, \ldots, S_k) \le \alpha(1+\epsilon)\text{OPT}(G_w, k)$. If $V' \ne V$ (i.e., if some edges e were contracted since $w(e) > \lambda$), then the run time to recover an $\alpha(1+\epsilon)$-approximate minmax k-partition in G_w from $(S_1, \ldots, S_k)$ is $n^{O(1)}$. □

5 Proof of Theorem 2

In this section, we restate and prove Theorem 2.

Theorem 2 *There exists a randomized algorithm that takes as input a weighted instance of* MINMAX k-CUT*, namely an n-vertex simple graph $G = (V, E)$ with edge weights $w : E \to \mathbb{Z}_+$ and an integer $k \geq 2$, along with an $\epsilon \in (0, 1)$, and runs in time $(k/\epsilon)^{O(k^2)} n^{O(1)} \log^2(\sum_{e \in E} w(e))$ to return a partition $\mathcal{P}$ of the vertices of G such that $cost_{G_w}(\mathcal{P}) \leq (1 + \epsilon) OPT(G_w, k)$ with high probability.*

Proof Given the input, we compute a family $\mathcal{C}$ as described in Theorem 5. For each instance $H \in \mathcal{C}$, we use Theorem 3 with $\lambda = O((k/\epsilon^3) \log n)$ to find an optimum to the instance or obtain that the optimum is $\Omega((k/\epsilon^3) \log n)$. By conclusion (iii) of Theorem 5, we can then recover and output a k-partition $\mathcal{P}$ of V such that $\text{cost}_{G_w}(\mathcal{P}) = (1 + O(\epsilon))\text{OPT}(G_w, k)$ in time $n^{O(1)}$. The correctness probability comes from the correctness probability in Theorem 5.

By Theorem 5, the time required to compute the collection $\mathcal{C}$ is

$$\text{poly}(n, 1/\epsilon) \log^2 \left(\sum_{e \in E} w(e) \right).$$

For each $H \in \mathcal{C}$, the total time to run the algorithm described in Theorem 3 with $\lambda = O((k/\epsilon^3) \log n)$ is

$$\begin{aligned} \left(\left(\frac{k^2}{\epsilon^3} \right) \log n \right)^{O(k^2)} n^{O(1)} &= \left(\frac{k}{\epsilon^3} \right)^{O(k^2)} (\log n)^{O(k^2)} n^{O(1)} \\ &= \left(\frac{k}{\epsilon^3} \right)^{O(k^2)} (k^{O(k^2)} + n) n^{O(1)} \\ &= \left(\frac{k}{\epsilon} \right)^{O(k^2)} n^{O(1)}. \end{aligned}$$

Finally, our algorithm requires $n^{O(1)}$ time to recover a k-partition in G from an optimum k-partition of an instance $H \in \mathcal{C}$. Therefore, the total run time is

$$\begin{aligned} &\text{poly}(n, 1/\epsilon) \log^2 \left(\sum_{e \in E} w_e \right) + |\mathcal{C}| \cdot ((k/\epsilon)^{O(k^2)} n^{O(1)} + k n^{O(1)}/\epsilon + n^{O(1)}) \\ &= (k/\epsilon)^{O(k^2)} n^{O(1)} \cdot \text{poly}(n, 1/\epsilon) \log^2 \left(\sum_{e \in E} w(e) \right) \\ &= (k/\epsilon)^{O(k^2)} n^{O(1)} \log^2 \left(\sum_{e \in E} w(e) \right). \end{aligned}$$

□

6 NP-hardness

In this section, we restate and prove the hardness result.

Theorem 1 MINMAX k-CUT *is strongly NP-hard and W[1]-hard when parameterized by k.*

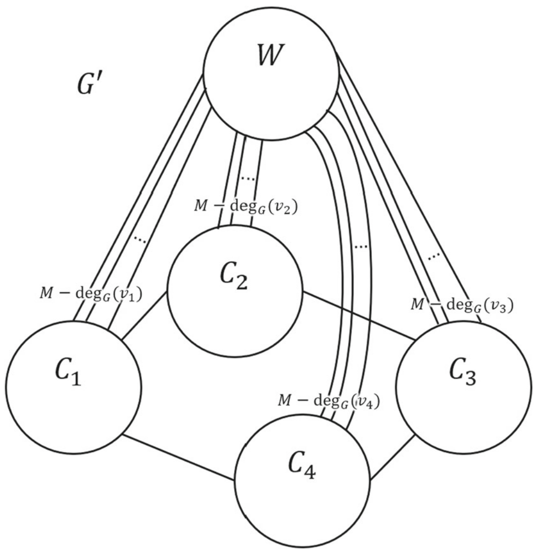

Fig. 6 The graph G' constructed in the Proof of Theorem 1 when the input graph G is a 4-cycle

Proof We will show a reduction from h- CLIQUE in the unweighted case. In h- CLIQUE, the input consists of a simple graph $G = (V, E)$ and a positive integer h, and the goal is to decide whether there exists a subset S of V of size $|S| = h$ with $G[S]$ being a clique. The h- CLIQUE problem is NP-hard and W[1]-hard when parameterized by h.

Let $(G = (V, E), h)$ be an instance of h- CLIQUE, where $V = \{v_1, \ldots, v_n\}$. We may assume that $h \geq 2$. Let $M := \max\{(n+1)^2, 3m\}$ and $N := Mn + 2$, where m is the number of edges in G. We construct a graph $G' = (V', E')$ as follows: for each vertex $v_i \in V$, we create a clique of size N over the vertex set $C_i := \{u_{i,j} : j \in [N]\}$. We also create a clique of size N over the vertex set $W := \{w_j : j \in [N]\}$. For each edge $e = v_i v_{i'} \in E$, we add the edge $u_{i,1}u_{i',1}$ (between the first copy of vertex v_i and the first copy of vertex $v_{i'}$). For each $v_i \in V$, we also add $M - \deg_G(v_i)$ edges between arbitrary pair of vertices in C_i and W—for the sake of clarity, we fix these $M - \deg_G(v_i)$ edges to be $u_{i,j}w_j$ for every $j \in [M - \deg_G(v_i)]$. We set $V' := W \cup \cup_{i\in[n]} C_i$. We note that the size of the graph G' is polynomial in the size of the input graph G. See Fig. 6 for an example. The next claim completes the reduction. □

Claim 8 *The graph G contains an h-clique if and only if there exists an* $(h+1)$*-partition* $\mathcal{P} = (P_1, \ldots, P_{h+1})$ *of* V' *such that* $\max_{i\in[h+1]} |\delta_{G'}(P_i)| \leq Mh - h(h-1)$.

Proof Suppose that G contains an h-clique induced by a subset $V_0 \subseteq V$. We may assume that $V_0 = \{v_1, \ldots, v_h\}$ by relabelling the vertices of V. Consider the partition $\mathcal{P} = (P_1, \ldots, P_{h+1})$ given by

$$P_i := C_i \ \forall i \in [h] \text{ and}$$
$$P_{h+1} := V' \backslash (P_1 \cup \ldots \cup P_h).$$

We observe that

$$\begin{aligned}|\delta_{G'}(P_i)| &= (M - \deg_G(v_i)) + \deg_G(v_i) = M \text{ for all } i \in [h] \text{ and}\\ |\delta_{G'}(P_{h+1})| &= \sum_{v\in V_0}(M - \deg_G(v)) + |E[V_0, V\backslash V_0]|\\ &= \sum_{v\in V_0}(M - \deg_G(v)) + \sum_{v\in V_0}\deg_G(v) - 2|E[V_0]|\\ &= Mh - 2|E[V_0]|\\ &= Mh - h(h-1).\end{aligned}$$

By choice of M, we know that $M \leq Mh - h(h-1)$, and hence $\max_{i\in[h+1]}|\delta_{G'}(P_i)| = Mh - h(h-1)$.

We now prove the converse. Suppose that we have an $(h+1)$-partition $\mathcal{P} = (P_1, \ldots, P_{h+1})$ of V' such that $\max_{i\in[h+1]}\{|\delta_{G'}(P_i)|\} \leq Mh - h(h-1)$.

We now show that $\mathcal{P}$ does not separate any two vertices in C_i for all $i \in [n]$ or any two vertices in W. For the sake of contradiction, suppose that there exists vertices $u, v \in V'$ such that u and v are in the same set $A \in \{C_1, \ldots, C_n, W\}$ but they are in different parts of $\mathcal{P}$. Without loss of generality, let $u \in P_1$ and $v \in P_2$. Then $A \cap P_1$ and $A\backslash P_1$ together forms a non-trivial 2-cut of $G'[A]$, and hence $|\delta_{G'}(P_1)| \geq |\delta_{G'[A]}(A\cap P_1)| \geq N-1$. By our choice of N, we know that $N-1 > Mh - h(h-1)$. This contradicts the fact that $|\delta_{G'}(P_i)| \leq Mh - h(h-1)$ for all $i \in [h+1]$.

From now on, let us fix P_{h+1} to be the part of $\mathcal{P}$ that contains W. We will show that P_{h+1} contains exactly $n-h$ sets among $C_1, \ldots, C_n$. Since all parts of $\mathcal{P}$ are non-empty, it follows that P_{h+1} cannot contain more than $n-h$ sets among $C_1, \ldots, C_n$. For the sake of contradiction, suppose that P_{h+1} contains at most $n-h-1$ sets among $C_1, \ldots, C_n$. This implies that more than h sets among $C_1, \ldots, C_n$ are not contained in P_{h+1}. Let $C_{i_1}, \ldots, C_{i_{h'}}$ be the sets outside P_{h+1}, where $h' > h$. Then

$$\begin{aligned}|\delta_{G'}(P_{h+1})| &\geq \sum_{\ell\in[h']}|E'[C_{i_\ell}, W]|\\ &= \sum_{\ell\in[h']}(M - \deg_G(v_{i_\ell}))\\ &\geq \sum_{\ell\in[h']}(M-n)\\ &\geq (h+1)(M-n)\\ &> Mh - h(h-1).\end{aligned}$$

This contradicts the fact that $|\delta_{G'}(P_i)| \leq Mh - h(h-1)$ for all $i \in [h+1]$. Hence, P_{h+1} contains exactly $n-h$ sets among $C_1, \ldots, C_n$.

Let $C_{i_1}, \ldots, C_{i_h}$ be the sets that are not in P_{h+1}. We will now show that $S := \{v_{i_1}, \ldots, v_{i_h}\}$ induces a clique in G. Since $\max_{i\in[h+1]}|\delta_{G'}(P_i)| \leq Mh - h(h-1)$, we know that

$$\begin{aligned} Mh - h(h-1) &\geq |\delta_{G'}(P_{h+1})| \\ &= \sum_{\ell \in [h]} (M - \deg_G(v_{i_\ell})) + |E'[\cup_{\ell \in [h]} C_{i_\ell}, (\cup_{i \in [n]} C_i) \backslash (\cup_{\ell \in [h]} C_{i_\ell})]| \\ &= \sum_{\ell \in [h]} (M - \deg_G(v_{i_\ell})) + |E[S, V \backslash S]| \\ &= Mh - \sum_{\ell \in [h]} \deg_G(v_{i_\ell}) + |E[S, V \backslash S]| \\ &= Mh - 2|E[S]|. \end{aligned}$$

Consequently, $|E[S]| \geq h(h-1)/2$, and thus S induces an h-clique in G. □

We remark that our hardness reduction also implies that an exact algorithm for MINMAX k-CUT in $n^{o(k)}$ time in simple graphs (i.e., unweighted graphs) would also imply an $n^{o(h)}$-time algorithm for h- CLIQUE, thus refuting the exponential time hypothesis (see Theorem 14.21 in [7]).

7 Conclusion

Our work adds to the exciting recent collection of works aimed at improving the algorithmic understanding of alternative objectives in graph partitioning [1, 5, 17]. We addressed the graph k-partitioning problem under the minmax objective. We showed that it is NP-hard, W[1]-hard when parameterized by k, and admits a parameterized approximation scheme when parameterized by k. Our algorithmic ideas generalize in a natural manner to also lead to a parameterized approximation scheme for MIN ℓ_p-NORM k-CUT for every $p \geq 1$: in MIN ℓ_p-NORM k-CUT, the input is a graph $G = (V, E)$ with edge weights $w : E \to \mathbb{Z}_+$, and the goal is to partition the vertices into k non-empty parts $V_1, V_2, \ldots, V_k$ so as to minimize $(\sum_{i=1}^{k} w(\delta(V_i))^p)^{1/p}$. We note that MIN ℓ_p-NORM k-CUT generalizes MINSUM k-CUT as well as MINMAX k-CUT.

Based on prior works in approximation literature for minmax and minsum objectives, it is a commonly held belief that the minmax objective is harder to approximate than the minsum objective. Our results suggest that for the graph k-partitioning problem, the complexity/approximability of the two objectives are perhaps the same. A relevant question towards understanding if the two objectives exhibit a complexity/approximability gap is the following: When k is part of input, is MINMAX k-CUT constant-approximable? We recall that when k is part of input MINSUM k-CUT does not admit a $(2-\epsilon)$-approximation for any constant $\epsilon > 0$ under the Small Set Expansion Hypothesis [24] and admits a 2-approximation [27]. The best approximation factor that we know currently for MINMAX k-CUT is $2k$ (see Sect. 1.1). A reasonable stepping stone would be to show that MINMAX k-CUT is APX-hard. We note that having a constant factor approximation for MINMAX k-CUT would immediately shave off the $\log \max_{e \in E} w(e)$ term from the run-time mentioned in Theorem 2.

The 2-approximation for MINSUM k-CUT is based on solving the same problem in the *Gomory-Hu* tree of the given graph. We note that solving MINMAX k-CUT on the Gomory-Hu tree of the given graph could at best result in an $O(n)$-approximation:

consider the complete graph K_{2n+1} on $2n+1$ vertices with unit edge weights. The optimum partition for MINMAX k-CUT for $k=n$ is the partition that contains $n-1$ parts each containing 2 vertices and one part containing 3 vertices leading to an optimum value of $3(2n-2)=\Theta(n)$. The star graph on $2n+1$ vertices with all edge weights being $2n$ is a Gomory-Hu tree for K_{2n+1}. The optimum partition for MINMAX k-CUT for $k=n$ on the Gomory-Hu tree (i.e., the star graph with weighted edges) consists of $n-1$ parts corresponding to $n-1$ leaves of the star graph and one part containing the remaining leaves and the center, thus leading to an optimum value of $\Theta(n^2)$.

Funding Supported in part by NSF grants CCF-1814613 and CCF-1907937.

References

1. Ahmadi, S., Khuller, S., Saha, B.: Min-max correlation clustering via multicut, pp. 13–26. Integer Programming and Comb. Optim, IPCO (2019)
2. Bansal, N., Feige, U., Krauthgamer, R., Makarychev, K., Nagarajan, V., Naor, J., Schwartz, R.: Min-max graph partitioning and small set expansion. SIAM J. Comput. **43**(2), 872–904 (2014)
3. Bérczi, K., Chandrasekaran, K., Király, T., Madan, V.: Improving the Integrality Gap for Multiway Cut. Math. Programming **183**, 171–193 (2020)
4. Chandrasekaran, K., Chekuri, C.: Min-max partitioning of Hypergraphs and Symmetric Submodular Functions. In: Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms, SODA, pp. 1026–1038 (2021)
5. Charikar, M., Gupta, N., Schwartz, R.: Local guarantees in graph cuts and clustering, pp. 136–147. Integer Programming Comb. Optim, IPCO (2017)
6. Chekuri, C., Quanrud, K., Xu, C.: LP Relaxation and Tree Packing for Minimum k-Cut. SIAM J. Discrete Math. **34**(2), 1334–1353 (2020)
7. Cygan, M., Fomin, F., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer, Cham (2015)
8. Cygan, M., Komosa, P., Lokshtanov, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S., Wahlstrom, M.: Randomized contractions meet lean decompositions. arXiv: https://arxiv.org/abs/1810.06864 (2018)
9. Dahlhaus, E., Johnson, D., Papadimitriou, C., Seymour, P., Yannakakis, M.: The complexity of multi-terminal cuts. SIAM J. Comput. **23**(4), 864–894 (1994)
10. Downey, R., Estivill-Castro, V., Fellows, M., Prieto, E., Rosamund, F.: Cutting up is hard to do: The parameterised complexity of k-cut and related problems. Electron. Notes Theor. Comput. Sci. **78**, 209–222 (2003)
11. Goldschmidt, O., Hochbaum, D.: Polynomial algorithm for the k-cut problem. In: Proceedings of the 29th Annual Symposium on Foundations of Computer Science, FOCS, pp. 444–451 (1988)
12. Goldschmidt, O., Hochbaum, D.: A Polynomial Algorithm for the k-cut Problem for Fixed k. Math. Oper. Res. **19**(1), 24–37 (1994)
13. Gupta, A., Harris, D., Lee, E., Li, J.: Optimal Bounds for the k-cut Problem. arXiv:2005.08301 (2020)
14. Gupta, A., Lee, E., Li, J.: An FPT Algorithm Beating 2-Approximation for k-Cut. In: Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, pp. 2821–2837 (2018)
15. Gupta, A., Lee, E., Li, J.: Faster exact and approximate algorithms for k-cut. In: Proceedings of the 59th IEEE annual Symposium on Foundations of Computer Science, FOCS, pp. 113–123 (2018)
16. Gupta, A., Lee, E., Li, J.: The Karger-Stein Algorithm is Optimal for k-Cut. In: Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC, p. 473-484 (2020)
17. Kalhan, S., Makarychev, K., Zhou, T.: Correlation clustering with local objectives. Adv. Neural Inform. Process. Syst. **32**, 9346–9355 (2019)
18. Karger, D.: Random sampling in cut, flow, and network design problems. Math. Oper. Res. **24**(2), 383–413 (1999)
19. Karger, D., Stein, C.: A new approach to the minimum cut problem. J. ACM **43**(4), 601–640 (1996)
20. Kawarabayashi, K. I., Lin, B.: A nearly 5/3-approximation FPT Algorithm for Min-k-Cut. In: Proceedings of the 31st ACM-SIAM Symposium on Discrete Algorithms, SODA, pp. 990–999 (2020)
21. Lawler, E.: Cutsets and Partitions of Hypergraphs. Networks **3**, 275–285 (1973)

22. Li, J.: Faster minimum k-cut of a simple graph. In: Proceedings of the 60th Annual Symposium on Foundations of Computer Science, FOCS, pp. 1056–1077 (2019)
23. Lokshtanov, D., Saurabh, S., Surianarayanan, V.: A Parameterized Approximation Scheme for MIN k-CUT. In: Proceedings of the 61st IEEE annual Symposium on Foundations of Computer Science, FOCS, pp. 798–809 (2020)
24. Manurangsi, P.: Inapproximability of Maximum Biclique Problems, Minimum k-Cut and Densest At-Least-k-Subgraph from the Small Set Expansion Hypothesis. Algorithms **11**(1), 10 (2018)
25. Narayanan, H., Roy, S., Patkar, S.: Approximation algorithms for min-k-overlap problems using the principal lattice of partitions approach. J. Algorithms **21**(2), 306–330 (1996)
26. Ravi, R., Sinha, A.: Approximating k-cuts using network strength as a lagrangean relaxation. Eur. J. Oper. Res. **186**(1), 77–90 (2008)
27. Saran, H., Vazirani, V.: Finding k Cuts within Twice the Optimal. SIAM J. Comput. **24**(1), 101–108 (1995)
28. Sharma, A., Vondrák, J.: Multiway cut, pairwise realizable distributions, and descending thresholds. In: Proceedings of the forty-sixth annual ACM Symposium on Theory of Computing, STOC, pp. 724–733 (2014)
29. Svitkina, Z., Tardos, É.: Min-max multiway cut. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX, pp. 207–218 (2004)
30. Thorup, M.: Minimum k-way Cuts via Deterministic Greedy Tree Packing. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC, pp. 159–166 (2008)
31. Zhao, L., Nagamochi, H., Ibaraki, T.: Greedy splitting algorithms for approximating multiway partition problems. Math. Programming **102**(1), 167–183 (2005)