Fixed Parameter Approximation Scheme for Min-Max k-Cut

Karthekeyan Chandrasekaran and Weihang Wang(✉)

University of Illinois Urbana-Champaign, Urbana, IL 61801, USA
{karthe,weihang3}@illinois.edu

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 edge weights $w : E \to \mathbb{R}_+$ 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 $(\lambda k)^{O(k^2)} n^{O(1)}$, where λ is the value of the optimum and n is the number of vertices. 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

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 edge weights $w : E \to \mathbb{R}_+$ along with an integer $k \geq 2$ and the goal is to partition the vertices of G into k non-empty parts $V_1, \ldots, V_k$ 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 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

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

M. Singh and D. P. Williamson (Eds.): IPCO 2021, LNCS 12707, pp. 354–367, 2021.
https://doi.org/10.1007/978-3-030-73879-2_25

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 [1,7,19]) 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 [22]. 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. The objectives in MINMAX k-CUT and MINSUM k-CUT for $k = 2$ 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 [13,14] 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 [20] 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 [8,10,15–18,23,25]. Very recently, Gupta, Harris, Lee, and Li [15,18] 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 [23]. 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 [26,31]. For k being a part of the input, Manurangsi [25] 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 $f(k)$ [9,12]. Motivated by this hardness result and Manurangsi's $(2-\epsilon)$-inapproximability result, Gupta, Lee, and Li [16] 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 $2^{O(k^6)}n^{O(1)}$ [16] and a

$(1+\epsilon)$-approximation algorithm that runs in time $(k/\epsilon)^{O(k)}n^{k+O(1)}$ [17]. Subsequently, Kawarabayashi and Lin [21] 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 [24] 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 edge weights $w: E \to \mathbb{R}_+$ 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 $k \geq 3$ [11] 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 [4]. 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 is a fundamental structural difference between Minmax k-cut and Minsum k-cut. Optimal solutions to Minsum k-cut satisfy a nice property: assuming that the input graph is connected, every part in an optimal partition for Minsum k-cut induces a connected subgraph. Hence, Minsum k-cut is also phrased as the problem of finding a minimum weight subset of edges to remove so that the resulting graph contains at least k connected components. However, this nice property does not hold for Minmax k-cut as illustrated by the example in Fig. 1.

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. 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 [15,18,20] (also see [8]). 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.[1] In an upcoming work, Chandrasekaran and Chekuri [5] show that the more general problem of min-max

[1] 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 $f(A)+f(B) \geq f(A \cap B)+f(A \cup B)$.

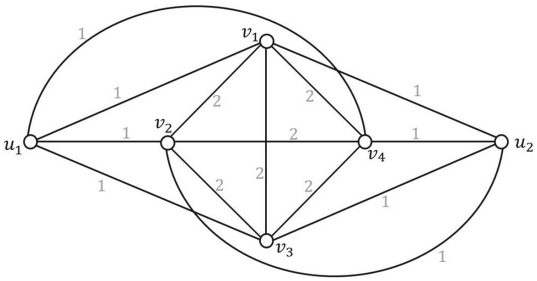

Fig. 1. An example where the unique optimum partition for MINMAX k-CUT for $k = 5$ induces a disconnected part. The edge weights are as shown. The unique optimum partition for MINMAX 5-CUT is $(\{u_1, u_2\}, \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\})$.

symmetric submodular k-partition[2] 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.

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 edge weights $w : E \to \mathbb{R}_+$. We write G to denote the unit-weight version of

[2] 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 $\max_{i=1}^{k} f(V_i)$.

the graph (i.e., the unweighted graph) and G_w to denote the graph with edge weights w. We emphasize that the unweighted graph could have parallel edges. 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 $\text{OPT}(G_w, k)$. 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 an instance of* Minmax k-cut*, namely an n-vertex graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}_{\geq 0}$ and an integer $k \geq 2$, along with an $\epsilon \in (0, 1)$, and runs in time $(k/\epsilon)^{O(k^2)} n^{O(1)} \log(\max_{e \in E} w(e))$ to return a partition $\mathcal{P}$ of the vertices of G with k non-empty parts such that $cost_{G_w}(\mathcal{P}) \leq (1 + \epsilon) OPT(G_w, k)$ with high probability.*

We note that $\log(\max_{e \in E} w(e))$ is polynomial in the size of the input. Theorem 2 can be viewed as the counterpart of the parameterized approximation scheme for Minsum k-cut due to Lokshtanov, Saurabh, and Surianarayanan [24] 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 graphs when parameterized by k and the solution size.

Theorem 3. *There exists a deterministic algorithm that takes as input an unweighted instance of* Minmax k-cut*, namely an n-vertex graph $G = (V, E)$ and an integer $k \geq 2$, along with an integer λ, and runs in time $(k\lambda)^{O(k^2)} n^{O(1)}$ 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, then finds an optimum.*

We emphasize that the algorithm in Theorem 3 is deterministic while the algorithm in Theorem 2 is randomized.

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 Downey et al.'s reduction [12] from the clique problem to Minsum k-cut.

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. Moreover, the reduction runs in time $2^{O(k)}(n/\epsilon)^{O(1)} \log \text{OPT}(G_w, k)$. Applying Theorem 3 to the reduced instance yields a run-time of

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

Hence, the total run-time is $(k/\epsilon)^{O(k^2)} n^{O(1)} \log \mathrm{OPT}(G_w, k)$ (including the reduction time), 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. (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 $\epsilon \mathrm{OPT}$ edges and hence, a $(1+\epsilon)$-approximate solution in the resulting graph gives a $(1+O(\epsilon))$-approximate solution in the original graph. (iii) Finally, we do a random sampling of edges with probability $q := \Theta(k \log n/(\epsilon^3 \mathrm{OPT}))$. This gives a subgraph that preserves all cut values within a $(1 \pm \epsilon)$-factor when scaled by q with high probability [3]. 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 q allows us to conclude that the optimum in the subsampled graph is $O((k/\epsilon^3)) \log n$. We note that this three step reduction recipe follows the ideas of [24] 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.

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. At a high-level, we exploit the tools developed by [24] 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 the minmax objective, the subproblems in our dynamic program are completely different from that of [24]. We begin with the observation that an optimum solution to Minmax k-cut is a k-approximate optimum to Minsum k-cut. This observation and the tree packing approach for Minsum k-cut due to [8] allows us to obtain, in polynomial time, a spanning tree T of the input graph such that the number of edges of the tree crossing a Minmax k-cut optimum partition is $O(k^2)$. We will call a partition Π with $O(k^2)$ edges of the tree T crossing Π to be a T-feasible partition. Next, we use the tools of [24] 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 [24]. We now sketch the sub-problems in our dynamic program. For simplicity, we assume that we have a value $\lambda \geq \mathrm{OPT}(G, k)$. We call a partition $\mathcal{P}$ of a set S to be a *k-subpartition* if $\mathcal{P}$ has at most k non-empty parts. The *adhesion* of a tree node t in the 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 $O(\lambda k)$ 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 k-subpartitions of the adhesion A_t and which can be extended to a T-feasible partition of the entire vertex set. A simple counting argument shows that $|\mathcal{F}^{A_t}| = (\lambda k)^{O(k^2)}$. Now consider a Boolean function $f_t : \mathcal{F}^{A_t} \times \{0, 1, \ldots, \lambda\}^k \times \{0, 1, \ldots, 2k^2\} \rightarrow \{0, 1\}$. We note that the domain of the function is small, i.e., the domain has size $(\lambda k)^{O(k^2)}$. Let $(\mathcal{P}_{A_t}, \bar{x}, d)$ denote an argument/input to the function. The function aims to determine if there exists a k-subpartition $\mathcal{P}$ 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 restriction of the partition $\mathcal{P}$ to A_t is exactly $\mathcal{P}_{A_t}$ (ii) the number of edges crossing 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 $\mathcal{P}$ whose both endpoints are in V_t is at most d. It is easy to see that if we can compute such a function f_r for the root node r of the tree decomposition, then it can be used to find the optimum value of MINMAX k-CUT, namely $\mathrm{OPT}(G, k)$.

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. On the one hand, given an arbitrary optimal partition Ω to MINMAX k-CUT, restricting Ω to V_t yields a partition $\mathcal{P}$ that satisfies (i) (ii), and (iii) above for some choice of $\mathcal{P}_{A_t}$, $\bar{x}$, and d. On the other hand, a partition $\mathcal{P}$ of V_t satisfying (i) (ii), and (iii) for some choice of $\mathcal{P}_{A_t}$, $\bar{x}$, and d does not necessarily extend to an optimal partition of V. Therefore, identifying all inputs $(\mathcal{P}_{A_t}, \bar{x}, d)$ for which there exists a partition satisfying (i) (ii), and (iii) is not necessary for our purpose. Instead, for a fixed optimal partition Ω, it suffices if our f_t function evaluates to 1 on inputs $(\mathcal{P}_{A_t}, \bar{x}, d)$ for which (i) (ii), and (iii) are satisfied by the partition obtained by restricting Ω to V_t.

To identify partitions of V_t that potentially extend to Ω, for tree node t with bag $\chi(t)$, we construct a family $\mathcal{D}$ of *nice decompositions* such that $\mathcal{D}$ is small-sized (i.e., with $|\mathcal{D}| = (\lambda k)^{O(k^2)} n^{O(1)}$). A nice decomposition is a triple of the form $(O, \mathcal{P}_{\chi(t)}, \mathcal{Q}_{\chi(t)})$ satisfying certain properties (which are slightly different from the properties used in [24]). Here, O is a subset of $\chi(t)$ and $\mathcal{P}_{\chi(t)}, \mathcal{Q}_{\chi(t)}$ are partitions of $\chi(t)$, where $\mathcal{Q}_{\chi(t)}$ refines $\mathcal{P}_{\chi(t)}$. The constructed family $\mathcal{D}$ is such that at least one of the $\mathcal{Q}_{\chi(t)}$ partitions in $\mathcal{D}$ is a refinement of a restriction of Ω to $\chi(t)$. By induction hypothesis, we know that $f_{t'}(\mathcal{P}_{A_{t'}}, \bar{x}', d') = 1$ for all children t' of t and all inputs $(\mathcal{P}_{A_{t'}}, \bar{x}', d')$ for which the restriction of Ω to $V_{t'}$ satisfies (i) (ii), and (iii). In order to set $f_t(\mathcal{P}_{A_t}, \bar{x}, d) = 1$ for all inputs $(\mathcal{P}_{A_t}, \bar{x}, d)$ for which the restriction of Ω to V_t satisfies (i) (ii), and (iii), it suffices to consider all inputs $(\mathcal{P}_{A_t}, \bar{x}, d)$ for which there exists a partition $\mathcal{P}$ of V_t satisfying (i) (ii), and (iii) such that (a) the restriction of $\mathcal{P}$ to $\chi(t)$ coarsens $\mathcal{Q}_{\chi(t)}$ for some $\mathcal{Q}_{\chi(t)}$ in $\mathcal{D}$ and (b) for all children t' of t, the restriction of $\mathcal{P}$ to $\chi(t')$ gives a partition satisfying (i) (ii), and (iii) for some input $(\mathcal{P}_{A_{t'}}, \bar{x}', d')$. Therefore, the family $\mathcal{D}$ of nice decompositions and the $f_{t'}$ values for all children t' of t together suffice to identify all inputs $(\mathcal{P}_{A_t}, \bar{x}, d)$ that correspond to a restriction of Ω to V_t satisfying (i) (ii), and (iii). To expedite this process, for each nice decomposition $(O, \mathcal{P}_{\chi(t)}, \mathcal{Q}_{\chi(t)})$, we use the set O and the partition $\mathcal{P}_{\chi(t)}$ to limit

the number of coarsenings of $\mathcal{Q}_{\chi(t)}$ to be considered. During this process, every time $f_t(\mathcal{P}_{A_t}, \bar{x}, d)$ is set to 1 for some input $(\mathcal{P}_{A_t}, \bar{x}, d)$, we can only guarantee that there is an actual partition of V_t satisfying (i) (ii), and (iii) (but that partition is not necessarily the restriction of Ω to V_t). However, if the restriction of Ω to V_t indeed satisfies (i) (ii), and (iii) for some input $(\mathcal{P}_{A_t}, \bar{x}, d)$, then the process will indeed set $f_t(\mathcal{P}_{A_t}, \bar{x}, d)$ to 1.

To formalize our tolerance of this one-sided error in the dynamic program, we define the notion of f-correctness and f-soundness for the function f_t (see Definition 6 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 $(\lambda k)^{O(k^2)} n^{O(1)}$ (see Lemma 4). 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 $(\lambda k)^{O(k^2)} n^{O(1)}$, thus proving Theorem 3.

One of our main contributions beyond the techniques of [24] is the introduction of the clean notions of f-correctness and f-soundness in the sub-problems of the dynamic program and combining them to address the minmax objective. An advantage of our dynamic program (in contrast to that of [24]) 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 for every $p \geq 1$. This result 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 $p \geq 1$.

Organization. We set up the tools to prove Theorem 3 in Sect. 2. We sketch a proof of Theorem 3 in Sect. 3. We refer the reader to the full version of this work [6] for all missing proofs (including Theorems 1 and 2). We conclude with a few open questions in Sect. 4.

2 Tools for the Fixed-Parameter Algorithm

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 $\chi : V(\tau) \rightarrow 2^V$ 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 $r \in V(\tau)$. For a tree node $t \in V(\tau) \backslash \{r\}$, 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 r, we define $A_r := \emptyset$. 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). 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 $G[S]$ either has large cut value or one side of the partition is small in size. With these definitions, we have the following result from [24].

Lemma 1. *[24] There exists a polynomial time algorithm that takes a graph $G = (V, E)$, an integer $k \geq 2$, and an integer λ as input 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.

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 shows that we can efficiently find a spanning tree T of a given graph such that there exists an optimum k-partition that $2k^2$-respects T. It follows from Lemma 7 of [8] and the observation that an optimum solution to MINMAX k-CUT is a k-approximate optimum to MINSUM k-CUT.

Lemma 2. *[8] There exists a polynomial time algorithm that takes a graph G as input and returns a spanning tree T of G such that there exists an optimum min-max k-partition Π that $(2k^2)$-respects T.*

The next definition allows us to handle partitions of subsets that are crossed by a spanning tree at most $2k^2$ times. For $S \subseteq U$ and a partition/k-subpartition $\mathcal{P}$ of U, 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 $\mathcal{P}$.

Definition 5. *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) The restriction of $\mathcal{P}'$ to X is $\mathcal{P}$, 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.

3 Fixed-Parameter Algorithm Parameterized by k and Solution Size

In this section we prove Theorem 3. 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. 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 $\lambda \geq \mathrm{OPT}$. We will design a dynamic programming algorithm that runs in time $(\lambda k)^{O(k^2)} n^{O(1)}$ 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 Lemma 2 to obtain 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, and moreover T is a subgraph of G. 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.

Our algorithm is based on dynamic program (DP). We now state the subproblems in our dynamic program (DP), bound the number of subproblems in the DP, and prove Theorem 3. 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 3. *For every tree node $t \in V(\tau)$, we have $|\mathcal{F}^{A_t}| = (\lambda k)^{O(k^2)}$. Moreover, the collection $\mathcal{F}^{A_t}$ can be enumerated in $(\lambda k)^{O(k^2)}$ time.*

The proof of the lemma appears in the full version [6]. The following definition will be useful in identifying the subproblems of the DP.

Definition 6. *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 $\mathcal{P}_{A_t} = (\tilde{P}_1, \ldots, \tilde{P}_{k'}) \in \mathcal{F}^{A_t}$, $\bar{x} \in \{0, 1, \ldots, \lambda\}^k$, 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-sound *if for all $\mathcal{P}_{A_t} = (\tilde{P}_1, \ldots, \tilde{P}_{k'}) \in \mathcal{F}^{A_t}$, $\bar{x} \in \{0, 1, \ldots, \lambda\}^k$ 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. We crucially need distinct correctness and soundness definitions in our sub-problems in order for Bellman's principle of optimality to hold.

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. Its proof appears in the full version [6].

Proposition 1. *If we have a function $f_r : \mathcal{F}^{A_r} \times \{0, 1, \ldots, \lambda\}^k \times \{0, 1, \ldots, 2k^2\} \to \{0, 1\}$ that is both f-correct and f-sound, where r is the root of the tree decomposition τ, then*

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

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

By Proposition 1, it suffices to compute an f-correct and f-sound function f_r, where r is the root of the tree decomposition τ. The next technical lemma allows us to compute this in a bottom-up fashion on the tree decomposition.

Lemma 4. *There exists an algorithm that takes as input (τ, χ), a tree node $t \in V(\tau)$, Boolean functions $f_{t'} : \mathcal{F}^{A_{t'}} \times \{0, 1, \ldots, \lambda\}^k \times \{0, 1, \ldots, 2k^2\} \to \{0, 1\}$ 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 $f_t : \mathcal{F}^{A_t} \times \{0, 1, \ldots, \lambda\}^k \times \{0, 1, \ldots, 2k^2\} \to \{0, 1\}$ that is f-correct and f-sound.*

The proof of Lemma 4 is the most technical part of our contribution. Its proof appears in the full version [6]. We now complete the proof of Theorem 3 using Lemma 4 and Proposition 1.

Proof (Proof of Theorem 3). In order to compute a function $f_r : \mathcal{F}^{A_r} \times \{0, 1, \ldots, \lambda\}^k \times \{0, 1, \ldots, 2k^2\} \to \{0, 1\}$ that is both f-correct and f-sound, we can apply Lemma 4 on each tree node $t \in V(\tau)$ in a bottom up fashion starting from leaf nodes of the tree decomposition. Therefore, using Lemmas 3 and 4, the total run time to compute f_r is

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

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

4 Conclusion

Our work adds to the exciting recent collection of works aimed at improving the algorithmic understanding of alternative objectives in graph partitioning [1,7,19]. We addressed the graph k-partitioning problem under the minmax objective. 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 $p \geq 1$.

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 [25] and admits a 2-approximation [27]. The 2-approximation for Minsum k-cut is based on solving the same problem in the *Gomory-Hu* tree of the given graph. We are aware of examples where this approach for Minmax k-cut only leads to a $\Theta(n)$-approximation, where n is the number of vertices in the input graph (see full version [6]). 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.

References

1. Ahmadi, S., Khuller, S., Saha, B.: Min-max correlation clustering via multicut. In: Integer Programming and Combinatorial Optimization, pp. 13–26. IPCO (2019)
2. Bansal, N., et al.: Min-max graph partitioning and small set expansion. SIAM J. Comput. **43**(2), 872–904 (2014)
3. Benczúr, A., Karger, D.: Randomized approximation schemes for cuts and flows in capacitated graphs. SIAM J. Comput. **44**(2), 290–319 (2015)
4. Bérczi, K., Chandrasekaran, K., Király, T., Madan, V.: Improving the Integrality Gap for Multiway Cut. Mathematical Programming (2020)
5. Chandrasekaran, K., Chekuri, C.: Min-max partitioning of hypergraphs and symmetric submodular functions. In: Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms (to appear). SODA (2021)
6. Chandrasekaran, K., Wang, W.: Fixed Parameter Approximation Scheme for Min-max k-cut. arXiv: https://arxiv.org/abs/2011.03454 (2020)
7. Charikar, M., Gupta, N., Schwartz, R.: Local guarantees in graph cuts and clustering. In: Integer Programming and Combinatorial Optimization, pp. 136–147. IPCO (2017)
8. Chekuri, C., Quanrud, K., Xu, C.: LP relaxation and tree packing for minimum k-cuts. In: 2nd Symposium on Simplicity in Algorithms, pp. 7:1–7:18. SOSA (2019)
9. Cygan, M., et al.: Parameterized Algorithms. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3
10. Cygan, M., et al.: Randomized contractions meet lean decompositions. arXiv: https://arxiv.org/abs/1810.06864 (2018)
11. Dahlhaus, E., Johnson, D., Papadimitriou, C., Seymour, P., Yannakakis, M.: The complexity of multiterminal cuts. SIAM J. Comput. **23**(4), 864–894 (1994)
12. 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)
13. Goldschmidt, O., Hochbaum, D.: Polynomial algorithm for the k-cut problem. In: Proceedings of the 29th Annual Symposium on Foundations of Computer Science, pp. 444–451. FOCS (1988)
14. Goldschmidt, O., Hochbaum, D.: A polynomial algorithm for the k-cut problem for fixed k. Math. Oper. Res. **19**(1), 24–37 (1994)
15. Gupta, A., Harris, D., Lee, E., Li, J.: Optimal Bounds for the k-cut Problem. arXiv: https://arxiv.org/abs/2005.08301 (2020)
16. 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, pp. 2821–2837. SODA (2018)
17. 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, pp. 113–123. FOCS (2018)
18. 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, pp. 473–484. STOC (2020)
19. Kalhan, S., Makarychev, K., Zhou, T.: Correlation clustering with local objectives. Adv. Neural Inform. Process. Syst. **32**, 9346–9355 (2019)
20. Karger, D., Stein, C.: A new approach to the minimum cut problem. J. ACM **43**(4), 601–640 (1996)

21. 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, pp. 990–999. SODA (2020)
22. Lawler, E.: Cutsets and partitions of hypergraphs. Networks **3**, 275–285 (1973)
23. Li, J.: Faster minimum k-cut of a simple graph. In: Proceedings of the 60th Annual Symposium on Foundations of Computer Science, pp. 1056–1077. FOCS (2019)
24. 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 (to appear). FOCS (2020)
25. 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)
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, pp. 724–733. STOC (2014)
29. Svitkina, Z., Tardos, É.: Min-max multiway cut. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, pp. 207–218. APPROX (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, pp. 159–166. STOC (2008)
31. Zhao, L., Nagamochi, H., Ibaraki, T.: Greedy splitting algorithms for approximating multiway partition problems. Math. Program. **102**(1), 167–183 (2005)