Graph Partitioning with a Move Budget

Majid Behbahani[*] Morgan Stanley
Mina Dalirrooyfard Morgan Stanley
Elaheh Fata Queen's University
Yuriy Nevmyvaka Morgan Stanley

Abstract

In many real world networks, there already exists a (not necessarily optimal) k-partitioning of the network. Oftentimes, one aims to find a k-partitioning with a smaller cut value for such networks by moving only a few nodes across partitions. The number of nodes that can be moved across partitions is often a constraint forced by budgetary limitations. Motivated by such real-world applications, we introduce and study the r-move k-partitioning problem, a natural variant of the Multiway cut problem. Given a graph, a set of k terminals and an initial partitioning of the graph, the r-move k-partitioning problem aims to find a k-partitioning with the minimum-weighted cut among all the k-partitionings that can be obtained by moving at most r non-terminal nodes to partitions different from their initial ones. Our main result is a polynomial time $3(r+1)$ approximation algorithm for this problem. We further show that this problem is $W[1]$-hard, and give an FPTAS for when r is a small constant.

1 INTRODUCTION

Graph partitioning problems are among the most fundamental and widely used graph problems in the fields of artificial intelligence, theoretical computer science, operations research and operations management. A *k-partitioning* of a graph G refers to partitioning the graph's nodes into k-disjoint sets, clusters, or partitions. Oftentimes, the goal in partitioing problems is to find a k-partitioning with the least *cut value* (also referred to as *cut size* in the literature), which is the sum of the weights of the edges that are between different partitions. Graph partitioning is widely used in machine learning, parallel computing, computer vision, VLSI design, political districting, epidemiology and many more areas (Dickinson et al., 2001; Hendrickson and Kolda, 2000; Joachims, 2003; Kahng et al., 2011; King et al., 2012; Shah et al., 2019).

Given a graph[*] G, an integer $k \geq 2$ and k terminals, the well-known *Multiway cut* problem asks to find a partitioning of the nodes of G into k clusters with the smallest cut value among all k-partitionings that have exactly one terminal in each partition (Dahlhaus et al., 1992). In this paper, we use the terms *partitions* and *clusters*, interchangeably. It is known that for $k > 2$, the Multiway cut problem is APX-hard (Dahlhaus et al., 1994).

We introduce and study a natural variant of the Multiway cut problem called the *r-move k-partitioning* problem. Suppose that we are given a graph G with an initial k-partitioning, an integer $k \geq 2$, k terminals and a *move parameter* $r \in \mathbb{N}$. The problem only considers k-partitionings for which at most r non-terminal nodes have been moved from their initial partitions to new clusters. The r-move k-partitioning problem asks to find a k-partitioning that minimizes the cut value over all the k-partitionings considered.

Numerous examples of the r-move k-partitioning problem can be seen in real-world networks. In fact, in many networks, the underlying graph already has a k-partitioning and this k-partitioning is not optimal, i.e., the cut value is not minimized. For instance, the edge weights in a network may change over time, resulting in the sub-optimality of the initial partitioning. However, in most applications, we cannot afford moving many nodes from their initial k-partitioning, and so the goal is to improve the cut value by moving only "a few" nodes.

There has been a long line of works providing approximation algorithms for the Multiway cut problem (Dahlhaus et al., 1994; Călinescu et al., 1998; Angelidakis et al., 2017; Saran and Vazirani, 1995; Manokaran et al., 2008), most of which use a simple linear programming (LP) relaxation of the problem called the *CKR LP*, in honor of the authors who introduced it, Călinescu, Karloff and Rabani (Călinescu et al., 1998). The best approximation factor for the Multiway cut problem is 1.2965 due to Sharma and Vondrák

[*]Authors are ordered alphabetically.

Proceedings of the 27[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

[*]We use the terms "graphs" and "networks" interchangeably.

(2014), who round the CKR LP. Interestingly, Manokaran et al. (2008) showed that assuming the unique games conjecture (Khot, 2002), if the integrality gap of the CKR LP is α, then the Multiway cut problem cannot be approximated better than α. Bérczi et al. (2020) showed that the integrality gap of the CKR LP is at least 1.20016, hence the best approximation factor for the Multiway cut problem lies between 1.20016 and 1.2965. Closing this gap is an interesting open problem.

Perhaps the closest known variant of the Multiway cut problem to the r-move k-partitioning problem is the *min s-t cut with at most r nodes*, also known as the *Min r-size s-t cut* problem (Zhang, 2016). This problem asks to find a minimum value s-t cut, where there are at most r nodes in the partition that contains terminal node s (i.e., the s-side). This problem is a special case of the $(r-1)$-move 2-partitioning problem, when in the initial partitioning all non-terminal nodes are in the partition that contains node t. By moving at most $r-1$ nodes to the s-side, the s-side will have size at most r.

Even though the Min r-size s-t cut problem is a variant of the s-t cut problem, which is polynomially solvable, the Min r-size s-t cut problem is NP-hard (Chen et al., 2016). Thus, the r-move 2-partitioning and the r-move k-partitioning problems are NP-hard. It is also known that the Min r-size s-t cut problem admits a *Fixed-Parameter Tractable (FPT)* solution with parameter r when the graph is unweighted (Lokshtanov and Marx, 2013) (see Section 2 for a definition of FPT problems). Intuitively, one might wonder if the r-move 2-partitioning problem and the Min r-size s-t cut problem are equivalent. We show that the r-move 2-partitioning problem is in fact W[1]-hard; thus, *not equivalent* to the Min r-size s-t cut problem. We resort to designing approximation algorithms for the r-move k-partitioning problem for all $k \geq 2$. Hence, the main theme of this paper is around the following question: *How well can we approximate r-move k-partitioning?*

1.1 Our Results

We provide a comprehensive study of the r-move k-partitioning problem. Recall that the r-move k-partitioning problem is NP-hard (for any constant $k \geq 2$), hence, there exists no exact polynomial time algorithm for it, unless P = NP. Throughout the paper we think of k as a constant.

First, note that when r is also a constant, there exist a simple $O((nk)^r)$ algorithm for the problem. This algorithm tries every combination of r' nodes, for all $1 \leq r' \leq r$, to be moved to any of the k partitions. It is natural to ask whether we can devise an FPT algorithm with parameter r for the r-move k-partitioning problem. Our first result shows that the answer to this question is negative.

Theorem 1.1 *The r-move k-partitioning problem with parameter r is W[1]-hard.*

In Theorem 1.1, we use a polynomial time reduction from the Densest r-Subgraph problem, where the complete proof can be found in the supplementary materials. It is well-known that the Densest r-Subgraph problem with parameter r is W[1]-hard (Cai, 2008).

Next, we give a $(1+\epsilon)$-approximation algorithm for the r-move k-partitioning problem with running time $f(r,\epsilon)O(n^2)$ for any $\epsilon > 0$, where $f(\cdot,\cdot)$ is a function of r and ϵ. When r is a constant, this gives us an FPTAS (with parameter ϵ) with running time $O(n^2/\epsilon^r)$, faster than the $O(n^r)$ brute-force algorithm, in the expense of an approximation factor.

Theorem 1.2 *Given any r-move k-partitioning instance graph G with n nodes and any constant $\epsilon > 0$, there exists a $(1+\epsilon)$-approximation algorithm for the r-move k-partitioning problem on G with running time $(\frac{2(k-1)(1+\epsilon)}{\epsilon})^r r! c n^2$, where c is a universal constant independent of r and ϵ.*

The FPTAS and the proof of Theorem 1.2 are provided in the full version of the paper (Behbahani et al., 2024). The running time of the algorithm in Theorem 1.2 grows fast as r grows, which leads us to the following question: Can we design an approximation algorithm with a running time polynomial in both n *and* r? Our main result focuses on answering this question. We give a polynomial time approximation algorithm with approximation factor at most $3(r+1)$. This is done by extending the CKR linear program to an LP for the r-move k-partitioning problem by adding a single linear constraint reflecting the move constraint of the problem. Our approximation algorithm is a simple rounding scheme for this LP, which makes our approach very practical. The main challenge is in proving the approximation guarantees of our rounding, which is given in Section 3.

Theorem 1.3 *Given a positive integer r, there exists a randomized algorithm for the r-move k-partitioning problem on any n-node m-edge graph G that with approximation factor $\frac{2k}{k-1}(r+1)$ and running time $T_{LP}(n,m)+O(mk)$, where $T_{LP}(n,m)$ is the running time of solving a linear program with $O(n)$ variables and $O(m)$ constraints.*

Firstly, note that we can de-randomize this algorithm in the expense of an additional factor of n in the running time (see Section 3.2). Secondly, note that unlike Theorem 1.2, the running time of our algorithm in Theorem 1.3 is independent of r. For $k=2$, we give a rounding scheme for our LP with approximation factor at most $r+1$, resulting in an $(r+1)$-approximation algorithm for the r-move 2-partitioning problem, as demonstrated in the full version of the paper (Behbahani et al., 2024).We further show that the

integrality gap of the LP used in Theorem 1.3 is at least $r + 1$, demonstrating that our rounding algorithm is tight within a constant factor.

Finally, as our approximation factor is dependent on r, we resort to bicriteria algorithms to make it constant. This algorithm is provided in the paper's full version (Behbahani et al., 2024).

Theorem 1.4 *For any $1/2 < \gamma < 1$, there exists a polynomial time $(\frac{1}{1-\gamma}, \frac{5}{2\gamma-1})$-approximation randomized algorithm for r-move k-partitioning problem, where the first criterion is the number of nodes moved and the second criterion is the cut value.*

We provide two remarks to better understand the applicability of our results. Firstly, we show in the supplementary materials that r-move k-partitioning without terminals is also W[1] hard and so even though for a lot of partitioning problems the "without terminals" version is easier, this is not the case for r-move k-partitioning. Our second remark is stated below. We include simple complementary numerical evaluations of our algorithm in Section 4.

Meaningful range for r: The range of r in which Theorems 1.3 and 1.2 provide meaningful results can be understood as follows. For the case of $k = 2$, the Multiway cut problem is tractable and so it is possible to compute an optimal multiway cut C for the input graph in polynomial time. Suppose that cut C is obtained by moving $\hat{r}$ nodes from their initial partitions. Then, Theorems 1.3 and 1.2 are meaningful only when $r < \hat{r}$. This is because if $r \geq \hat{r}$, then C is an optimal output for the r-move 2-partitioning problem. On the other hand, when $k > 2$, there is no polynomial time algorithm for the Multiway cut problem. Therefore, in this case, one can obtain a cut C whose cut value is within a constant approximation factor of an optimal multiway cut, using an approximation algorithm for the Multiway cut problem (for example the $\frac{3}{2}$-approximation algorithm of Călinescu et al. (1998)). Let $\hat{r}$ be the number of nodes that C moves from their initial partitions. Once again, Theorems 1.3 and 1.2 are useful only when $r < \hat{r}$.

1.2 Related Works

The Multiway cut problem without terminals is called the k-partitioning problem. While the k-partitioning problem is polynomially solvable for fixed k (Goldschmidt and Hochbaum, 1994), the Multiway cut problem is NP-hard for $k \geq 3$ (Dahlhaus et al., 1992). There exist many "budgeted" variants of the Multiway cut and k-partitioning problems in the literature, we name a few of these variants here.

The Min r-size s-t cut problem: First, recall that the Min r-size s-t cut problem is NP-hard, even though the "without terminals" version of this problem, that is, the Min r-size cut problem, is polynomially solvable (Armon and Zwick, 2006; Watanabe and Nakamura, 1987). The Min r-size cut problem asks to find a cut with the minimum value among all the cuts with one side having at most r nodes. The result of Armon and Zwick (2006) is based on the result of Karger and Stein (1996) which states that the number of cuts with value at most α times the value of the min cut is at most $n^{2\alpha}$. Such result does not exist for s-t cuts and, in fact, the number of min s-t cuts can be exponentially many*. The Min r-size s-t cut problem has been studied by other names such as the problem of "cutting at most r nodes by edge-cut with terminal" (Fomin et al., 2013; Lokshtanov and Marx, 2013). Moreover, Zhang (2014) gives an $O(\log(n))$-approximation algorithm for the Min r-size s-t cut problem using Räcke's tree decomposition method (Räcke, 2008). See the full version of the paper (Behbahani et al., 2024), for a table of comparison of the related variants of the r-move 2-partitioning problem and their associated algorithms.

Other variants: Most of the variants of the Multiway k-cut problem that we are aware of are for $k = 2$. Basically having a budget on the number of nodes in one side of the cut, and having terminals or no terminals in the graph makes up of most of these variants.

The exact version of the Min r-size s-t cut problem is called the *Min Er-Size s-t cut* problem (Feige et al., 2003), which asks to find an s-t cut with minimum cut value among all the cuts that have exactly r nodes in the s-side. The $(r, n-r)$-*cut* problem (Bonnet et al., 2015) is the above problem without any terminals. The $(r, n-r)$-cut problem is known to be W[1]-hard (Bonnet et al., 2015), and as a result the Min Er-Size s-t cut problem is also W[1]-hard. For both these problems, there is a randomized $O(r/\log(n))$-approximation algorithm due to Feige et al. (2003). These problems have been studied when parameterized by cut value as well (Fomin et al., 2013).

For $k > 2$, the most relevant variant of the Multiway k-cut problem is the k-balanced partitioning problem, which is proved to be APX-hard (Feldmann and Foschini, 2015). Other problems that seem to be indirectly relevant are the k-route cut problem (Chuzhoy et al., 2015) and the MinSBCC problem (Hayrapetyan et al., 2005).

Dinitz et al. (2022) study a variant of the min s-t cut problem with fairness considerations. They introduce the Demographic Fair Cut problem, which is informally defined as follows: Given a terminal node s and a labeling of nodes into various demographics, the goal is to find a minimum cut that disconnects at least a certain given fraction of each demographic from s. If all nodes of the graph

*Consider a graph which consists of $n/2$ paths of length 2 from terminal s to terminal t. An s-t cut has at least one edge from each path, and so the number of min s-t cuts is $2^{n/2}$.

belong to a single demographic, then this problem is equivalent to the Min r-size s-t cut problem. Thus, their $\log(n)$-approximation algorithm can also be applied to the Min r-size s-t cut problem, resulting in a $\min(r+1, \log(n))$-approximation algorithm for the Min r-size s-t cut problem. Whether a $\log(n)$-approximation algorithm exists for the r-move k-partitioning problem (either for $k = 2$ or $k > 2$) is an interesting open question that is outside the scope of this paper.

2 PRELIMINARIES

Let $G = (V, E)$ with weight function $c : E \to \mathbb{R}$, that is, an edge $(u, v) \in E$ has weight $c(u, v)$ or c_{uv}. We refer to an edge with endpoints u and v by (u, v) or uv. We only consider undirected graphs throughout this paper. Let $E(G)$ and $V(G)$ denote the edge set and node set of a graph G. A problem of size n with parameter r is said to be FPT if there is an algorithm that solves it in $f(r)O(n^c)$ time, where $f(\cdot)$ is an arbitrary function depending only on r and c is a constant. If any $W[1]$-complete problem is FTP, then FPT = $W[1]$ and every problem in $W[1]$ is FPT.

We formally define the r-move k-partitioning problem using the definition of a k-cut (also known as k-partitioning). Given a weighted graph $G = (V, E)$ with weight function $c : E \to \mathbb{R}$, a *k-cut* is a subset of edges $E' \subseteq E$ such that removing the set of edges E' from the graph results in another graph $G' = (V, E \backslash E')$ that has k connected components. We refer to these connected components that appear after removing edge set E' as *clusters* or *partitions*. The weight of a k-cut E' is the sum of the weight of the edges in E' and it is called the *cut value*. Any set of edges that introduce a cut in a graph, such as E', can be referred to as a *cut-set*.

Definition 2.1 (The r-move k-partitioning problem) *Let $G = (V, E)$ be a weighted graph with a weight function $c : E \to \mathbb{R}$, a set of terminals $S = \{s_1, \ldots, s_k\} \subseteq V$ and $|V| = n$. Suppose that G has a given initial partitioning, where $\ell_v \in \{1, \ldots, k\}$ denotes the initial partition that node v belongs to and for a terminal node s_i, we have $\ell_{s_i} = i$, for each $i \in \{1, \ldots, k\}$. The r-move k-partitioning problem asks to find a minimum-weighted k-cut $\ell^* : V \to \{1, \ldots, k\}$ such that $\ell^*_v \neq \ell_v$ for at most r non-terminal nodes v.*

The parameter r is referred to as the *move parameter*. The linear program we use throughout this paper is represented in Table 1 and will be referred to as LP 1. Suppose that k is the number of partitions of interest. For each $v \in V$, let $X_v = (X^1_v, \ldots, X^k_v)$ be a vector of size k of positive real decision variables in $[0, 1]$. To understand the role of decision variable X^i_v better, observe that if we were only considering integer solutions, then $X^i_v = 1$ would represent node v being in partition i, and constraint (C4) would ensure that each node v is assigned to exactly one partition. We define the distance between pairs of vertices u, v as $d(u, v) = \frac{1}{2}\sum_{i=1}^{k} |X^i_u - X^i_v|$. In case of integer solutions, if u and v are in the same partition, then $d(u, v) = 0$ and if they are in different partitions, $d(u, v) = 1$. This further motivates the objective function in LP 1. To represent $|X^i_u - X^i_v|$ in the LP, we define variables Y^i_{uv}'s for each $(u, v) \in E$, and add constraints (C2) and (C3) to make sure $Y^i_{uv} = |X^i_u - X^i_v|$. Finally, we call constraint (C7) *the move constraints*, which ensures that at most r nodes are moved from their initial partitions, in case of integer solutions. Note that LP 1 without constraint (C7) is the CKR LP.

Suppose X is a feasible, but not necessarily optimal, solution to LP 1. Notation $d_X(u, v) = \frac{1}{2}\sum_{i=1}^{k} |X^i_u - X^i_v|$ is used to better clarify that the distance function is calculated specifically for solution X. We use the notation $\bar{X}$ to denote the optimal solution of LP 1. We conclude this section with the following lemmas that will be later used in our proofs.

Lemma 2.1 *Consider a feasible solution X to LP 1. Given a number $0 \leq \lambda < 1$, if L is the set of nodes v for which $X^{\ell_v}_v < \lambda$, then $|L| < \frac{r}{1-\lambda}$.*

Proof. For each $v \in L$, we have $1 - X^{\ell_v}_v > 1 - \lambda$. Since X is a feasible solution of LP 1, $r \geq \sum_{v \in L}(1 - X^{\ell_v}_v)$. So $r > (1 - \lambda)|L|$, and $|L| < \frac{r}{1-\lambda}$. □

Lemma 2.2 *Let X be any feasible solution to LP 1. For any $i \in \{1, \ldots, k\}$ and $u, v \in V$ we have $d_X(u, v) \geq |X^i_u - X^i_v|$.*

Proof. By the triangle inequality we have $\sum_{j \neq i} |X^j_u - X^j_v| \geq |\sum_{j \neq i} X^j_u - \sum_{j \neq i} X^j_v| = |(1 - X^i_u) - (1 - X^i_v)|$, where the last equality used $\sum_j X^j_u = \sum_j X^j_v = 1$. Therefore, $d_X(u, v) = \frac{1}{2}(|X^i_u - X^i_v| + \sum_{j \neq i} |X^j_u - X^j_v|) \geq |X^i_u - X^i_v|$. □

3 ALGORITHMS

This section discusses Theorem 1.3. The rest of the results are presented in the supplementary materials and the paper's full version (Behbahani et al., 2024).

3.1 $3(r+1)$-Approximation Algorithm for Parametric r

In this section, we show a $\frac{2k}{k-1}(r+1)$-approximation algorithm for the r-move k-partitioning problem when $k > 2$. For $k > 2$, we have that $\frac{2k}{k-1} \leq 3$; therefore, our algorithm provides a $3(r+1)$-approximation guarantee for such k. Note that, this algorithm works for $k = 2$ as well; however, we already have an $(r+1)$-approximation algorithm for

Minimize:	$\sum_{(u,v)\in E} c_{uv} d(u,v)$		
Subject to:			
	$d(u,v) = \frac{1}{2}\sum_{i=1}^{k} Y_{uv}^i$	$\forall (u,v) \in E$	(C1)
	$Y_{uv}^i \geq X_u^i - X_v^i$	$\forall (u,v) \in E, \forall i \in \{1 \cdots k\}$	(C2)
	$Y_{uv}^i \geq X_v^i - X_u^i$	$\forall (u,v) \in E, \forall i \in \{1 \cdots k\}$	(C3)
	$\sum_{i=1}^{k} X_u^i = 1$	$\forall u \in V$	(C4)
	$X_{s_i} = \mathbf{e}_i$	$\forall s_i \in S$	(C5)
	$X_v^i \geq 0$	$\forall v \in V, \forall i \in \{1, \ldots, k\}$	(C6)
	$\sum_{v\in V}(1 - X_v^{\ell_v}) \leq r$		(C7)

Table 1: r-move k-partitioning LP (LP 1)

this case, as presented in the full version (Behbahani et al., 2024).

Algorithm 1 $\frac{2k}{k-1}(r+1)$-approximation algorithm for the r-move k-partitioning problem

$\bar{X} \leftarrow$ solution to LP 1.
$g = \frac{k-1}{k(r+1)}$.
ρ chosen uniformly at random from $(0, g)$.
$A \leftarrow \{\mathbf{z} = (\mathbf{z}^1, \ldots, \mathbf{z}^k) | \frac{1}{g}\mathbf{z}^i \in \mathbb{Z}_{\geq 0}$ for all $1 \leq i \leq k$ and $\sum_{i=1}^{k} \mathbf{z}^i < 1 + kg\}$.
for i from 1 to k **do**
  for $v \in V(G)$ **do**
    $\tilde{X}_v^i = g\lfloor \frac{\bar{X}_v^i + \rho}{g} \rfloor$.
  end for
end for
for $\mathbf{z} \in A$ **do**
  $H_{\mathbf{z}} \leftarrow \{v \in V(G) | \tilde{X}_v = \mathbf{z}\}$.
  if $H_{\mathbf{z}}$ contains a terminal s_i **then** ▷ $H_{\mathbf{z}}$ can contain at most one terminal
    $i_{\mathbf{z}}^* \leftarrow i$.
  else
    $i_{\mathbf{z}}^* \leftarrow \arg\max_{j=1}^{k} |\{v \in H_{\mathbf{z}} | \ell_v = j\}|$.
  end if
  Assign all the nodes in $H_{\mathbf{z}}$ to $i_{\mathbf{z}}^*$.
end for

The algorithm operates as follows: The optimal solution to LP 1 gets rounded in two phases. Let $g = \frac{k-1}{k(r+1)}$ and ρ be a real number chosen uniformly at random from the interval $(0, g)$. In the first phase, for each node v, we round the entries of $\bar{X}_v$ to obtain $\tilde{X}_v$ such that for each $1 \leq i \leq k$, $\tilde{X}_v^i$ is an integer multiple of g. More precisely, let $\tilde{X}_v^i = \lfloor \frac{\bar{X}_v^i + \rho}{g} \rfloor g$, i.e., $\tilde{X}_v^i$ is the largest multiple of g no larger than $\bar{X}_v^i + \rho$.

In the second phase of rounding, the algorithm puts all nodes that are rounded to the same vector in the same partition, see Algorithm 1 for a description of how this partition is chosen. To clarify the role of ρ, for two nodes $u, v \in V(G)$ if $d_{\bar{X}}(u,v)$ is small, then we want $\bar{X}_u$ and $\bar{X}_v$ be rounded to the same vector with high probability, i.e., $\tilde{X}_u = \tilde{X}_v$ with high probability. If we let $\rho = 0$ at all times, then u and v never get rounded to the same vector in the following case: For some small $\epsilon > 0$, let $\bar{X}_v^1 = g + \epsilon = \bar{X}_u^2$, $\bar{X}_v^2 = g - \epsilon = \bar{X}_u^1$ and $\bar{X}_v^i = \bar{X}_u^i$ for $i > 2$. Thus, $d_{\bar{X}}(u,v) = 2\epsilon$ is small. For this example and $\rho = 0$ we have $\tilde{X}_v^1 = \lfloor \frac{\bar{X}_v^1}{g} \rfloor g = \lfloor \frac{g+\epsilon}{g} \rfloor g = g$ and $\tilde{X}_u^1 = \lfloor \frac{\bar{X}_u^1}{g} \rfloor g = \lfloor \frac{g-\epsilon}{g} \rfloor g = 0$. In other words, even though $d_{\bar{X}}(u,v)$ is small $\bar{X}_u$ and $\bar{X}_v$ are rounded to different vectors with certainty. This is in fact true for any constant value of ρ and a random ρ is key to be able to round two nodes u and v that are very close to each other to the same vector, with high probability.

Lemma 3.1 *For each $v \in V(G)$ and each $i \in \{1, \ldots, k\}$ we have $|\tilde{X}_v^i - \bar{X}_v^i| < g$. Moreover, if for two nodes $u, v \in V(G)$ and an $i \in \{1, \ldots, k\}$ we have $\tilde{X}_v^i = \tilde{X}_u^i$, then $|\bar{X}_v^i - \bar{X}_u^i| < g$.*

Proof. If $\tilde{X}_v^i \geq \bar{X}_v^i$, then since $\tilde{X}_v^i \leq \bar{X}_v^i + \rho$ and $\rho < g$, we have that $|\tilde{X}_v^i - \bar{X}_v^i| < g$. If $\tilde{X}_v^i < \bar{X}_v^i$, then since $\tilde{X}_v^i = \lfloor \frac{\bar{X}_v^i + \rho}{g} \rfloor g$, we have $(\frac{\bar{X}_v^i + \rho}{g} - 1)g < \tilde{X}_v^i$, thus $\bar{X}_v^i + \rho < \tilde{X}_v^i + g$. Therefore, $|\tilde{X}_v^i - \bar{X}_v^i| < g - \rho < g$.

For the second part of the lemma, since both $\bar{X}_v^i + \rho$ and $\bar{X}_u^i + \rho$ are rounded down to the same value $\alpha = \tilde{X}_v^i$, they are both in the $[\alpha, \alpha + g)$ interval. Therefore, $\bar{X}_v^i$ and $\bar{X}_u^i$ are both in the interval $[\alpha - \rho, \alpha + g - \rho)$, concluding that $|\bar{X}_v^i - \bar{X}_u^i| < g$. □

By Lemma 3.1, we have $|\tilde{X}_v^i - \bar{X}_v^i| < g$ and by constraint (C4) in LP 1 we have $\sum_{i=1}^{k} \bar{X}_v^i = 1$, thus, $\sum_{i=1}^{k} \tilde{X}_v^i < 1 + kg$. Let A be the set of all k-sized vectors with entries being non-negative integer multiples of g such that the sum of the entries of each vector is at most $1 + kg$. By definition, each entry of $\tilde{X}_v$ is non-negative integer multiple of g and, as shown, the sum of entries of $\tilde{X}_v$ is at most $1 + kg$. Therefore, Algorithm 1 rounds any $\bar{X}_v$ to a vector in A.

Now we show that if $d_{\bar{X}}(u,v)$ is very small, then with high probability $\tilde{X}_u = \tilde{X}_v$. Note that for two nodes u and v with $\tilde{X}_u = \tilde{X}_v$, Algorithm 1 puts u and v in the same partition.

Lemma 3.2 *For any two distinct nodes $u, v \in V(G)$ such*

that $(u, v) \in E(G)$, the probability that Algorithm 1 puts (u, v) in the cut-set is at most $\frac{2}{g} d_{\bar{X}}(u, v)$.

Proof. If $d_{\bar{X}}(u, v) > \frac{g}{2}$, then $1 < \frac{2}{g} d_{\bar{X}}(u, v)$ and the lemma holds trivially. Assume that $d_{\bar{X}}(u, v) \leq \frac{g}{2}$. Observe that u and v are assigned to different partitions only if they are rounded to different vectors, that is, $\tilde{X}_u \neq \tilde{X}_v$. This happens if there exists an $i \in \{1, \ldots, k\}$ and an integer $1 \leq t \leq \frac{1}{g}$ such that tg is between $\bar{X}^i_v + \rho$ and $\bar{X}^i_u + \rho$. To measure this probability, let $i \in \{1, \ldots, k\}$ be fixed. Without loss of generality, suppose that $\bar{X}^i_v < \bar{X}^i_u$. Then, we need to compute the probability that

$$\bar{X}^i_v + \rho < tg \leq \bar{X}^i_u + \rho. \tag{1}$$

To see feasible values of ρ and t, we consider the following two cases: (i) $\lfloor \frac{\bar{X}^i_v}{g} \rfloor = \lfloor \frac{\bar{X}^i_u}{g} \rfloor$, and (ii) $\lfloor \frac{\bar{X}^i_v}{g} \rfloor < \lfloor \frac{\bar{X}^i_u}{g} \rfloor$. For the first case, suppose that $\lfloor \frac{\bar{X}^i_v}{g} \rfloor = \lfloor \frac{\bar{X}^i_u}{g} \rfloor = \hat{t}$. This means that $\hat{t}g \leq \bar{X}^i_v < \bar{X}^i_u < (\hat{t}+1)g$. Since $\rho < g$, the only feasible value for t is $\hat{t}+1$ and inequality (1) is equivalent to

$$0 < (\hat{t}+1)g - \bar{X}^i_u \leq \rho < (\hat{t}+1)g - \bar{X}^i_v < g. \tag{2}$$

The length of the $[(\hat{t}+1)g - \bar{X}^i_u, (\hat{t}+1)g - \bar{X}^i_v)]$ interval is $|\bar{X}^i_u - \bar{X}^i_v|$. Therefore, in case (i), with probability at most $\frac{1}{g}|\bar{X}^i_u - \bar{X}^i_v|$ there exists a t satisfying inequality (1).

For the second case, suppose that $\hat{t} = \lfloor \frac{\bar{X}^i_v}{g} \rfloor < \lfloor \frac{\bar{X}^i_u}{g} \rfloor$. Since $g > \frac{g}{2} > d_{\bar{X}}(u, v) \geq \bar{X}^i_u - \bar{X}^i_v$, we must have $\lfloor \frac{\bar{X}^i_u}{g} \rfloor = \hat{t}+1$. The only accepted values for t is $\hat{t}+1$ and $\hat{t}+2$, and inequality (1) is satisfied when ρ is in one of the following two intervals:

$$0 \leq \rho < (\hat{t}+1)g - \bar{X}^i_v \quad \text{or} \quad (\hat{t}+2)g - \bar{X}^i_u \leq \rho \leq g.$$

The sum of the lengths of these two intervals is $|\bar{X}^i_u - \bar{X}^i_v|$, so in case (ii) edge (u, v) is in the cut-set with probability at most $\frac{1}{g}|\bar{X}^i_u - \bar{X}^i_v|$. Putting cases (i) and (ii) together, any edge $(u, v) \in E(G)$ is in the cut-set with probability at most $\frac{1}{g}|\bar{X}^i_u - \bar{X}^i_v|$. Finally, since $d(u, v) = \frac{1}{2}\sum_{i=1}^{k} |\bar{X}^i_u - \bar{X}^i_v|$, the probability that edge (u, v) is in the cut-set is at most $\frac{2}{g} d_{\bar{X}}(u, v)$. □

Let $\mathbf{z}$ be a vector in A and $\mathbf{z}^i$ denote its $i-$th entry. Let $H_{\mathbf{z}}$ be the set of all nodes $v \in V(G)$ for which $\tilde{X}_v = \mathbf{z}$. The following lemma discusses the number of terminal nodes that $H_{\mathbf{z}}$ may contain.

Lemma 3.3 *For any vector $\mathbf{z} \in A$, there can be at most one terminal node in $H_{\mathbf{z}}$.*

Proof. Suppose there exists two distinct terminal nodes s_i and s_j in $H_{\mathbf{z}}$. By constraint (C5) of LP 1, we have $\bar{X}^i_{s_i} = 1$ and $\bar{X}^i_{s_j} = 0$, providing $|\bar{X}^i_{s_i} - \bar{X}^i_{s_j}| = 1$. On the other hand, by Lemma 3.1 we have $|\bar{X}^i_{s_i} - \bar{X}^i_{s_j}| < g < 1$. This is a contradiction and there can be at most one terminal in $H_{\mathbf{z}}$. □

If there exists a terminal node $s_i \in H_{\mathbf{z}}$, then Algorithm 1 puts all the nodes in $H_{\mathbf{z}}$ in partition i. Otherwise, the algorithm puts all the nodes in $H_{\mathbf{z}}$ in the partition where most nodes of $H_{\mathbf{z}}$ come from. Let $r_{\mathbf{z}} = \sum_{v \in H_{\mathbf{z}}} (1 - \bar{X}^{\ell_v}_v)$.

Lemma 3.4 *If $r \geq 1$, then for each vector $\mathbf{z} \in A$, there exists a partition $i_{\mathbf{z}}$ for which the number of nodes $v \in H_{\mathbf{z}}$ with $\ell_v \neq i_{\mathbf{z}}$ is less than $r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r}$. If there is a terminal node s_i in $H_{\mathbf{z}}$, then $i_{\mathbf{z}} = i$.*

Proof. Consider a vector $\mathbf{z} \in A$ and its associated set of nodes $H_{\mathbf{z}}$. Let B_i be the set of nodes $v \in H_{\mathbf{z}}$ with $\ell_v = i$, i.e., the set of nodes in $H_{\mathbf{z}}$ whose initial partition is i. Similarly, we define B_{-i} to be the set of nodes $v \in H_{\mathbf{z}}$ with $\ell_v \neq i$, that is, the set of nodes in $H_{\mathbf{z}}$ whose initial partition is not i. To prove this lemma, we consider the following two cases: (i) there exists a terminal node $s_i \in H_{\mathbf{z}}$, and (ii) $H_{\mathbf{z}}$ contains no terminal node.

In case (i), for a node $v \in H_{\mathbf{z}}$, it holds that $\tilde{X}^i_v = \tilde{X}^i_{s_i}$. So, by Lemma 3.1, we have $\bar{X}^i_v > \bar{X}^i_{s_i} - g = 1 - g$. Using the fact that for any $v \in V(G)$ it holds that $\sum_{i \in \{1,\ldots,k\}} \bar{X}^i_v = 1$, we have $r_{\mathbf{z}} = \sum_{v \in H_{\mathbf{z}}} (1 - \bar{X}^{\ell_v}_v) \geq \sum_{v \in B_{-i}} (1 - \bar{X}^{\ell_v}_v) \geq \sum_{v \in B_{-i}} \bar{X}^i_v > |B_{-i}|(1-g)$, providing that $|B_{-i}| < \frac{1}{1-g} r_{\mathbf{z}}$. Combining this with $g = \frac{k-1}{k(r+1)} < \frac{1}{1+r}$, we have $|B_{-i}| < r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r}$.

In case (ii), $H_{\mathbf{z}}$ contains no terminal node. Without loss of generality, suppose that for all $2 < i \leq k$ we have $|B_1| \leq |B_2| \leq |B_i|$. Defining $B' = H_{\mathbf{z}} \setminus (B_1 \cup B_2)$, we have $|B_1| \leq \frac{|B'|}{k-2}$. For the sake of contradiction, suppose for all $i \in \{1, \ldots, k\}$ we have $|B_{-i}| = |H_{\mathbf{z}} \setminus B_i| \geq r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r}$. In particular, $r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r} \leq |H_{\mathbf{z}} \setminus B_2| = |B'| + |B_1| \leq |B'| + \frac{|B'|}{k-2}$. So $|B'| \geq \frac{k-2}{k-1}(r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r})$. By Lemma 3.1, for any two nodes $v, u \in H_{\mathbf{z}}$, we have $|\bar{X}^{\ell_v}_v - \bar{X}^{\ell_v}_u| < g$; thus, $1 - \bar{X}^{\ell_v}_v > 1 - \bar{X}^{\ell_v}_u - g$. If $\ell_v \notin \{1, 2\}$, that is, $v \in B'$, then $\sum_{i \in \{1,\ldots,k\}, i \neq \ell_v} \bar{X}^i_v > g + \sum_{i \in \{1,2\}, i \neq \ell_v} (\bar{X}^i_u - g) + \sum_{i \in \{3,\ldots,k\}, i \neq \ell_v} \bar{X}^i_u$. If $\ell_v \in \{1, 2\}$, then we have $\sum_{i \in \{1,\ldots,k\}, i \neq \ell_v} \bar{X}^i_v > \sum_{i \in \{1,2\}, i \neq \ell_v} (\bar{X}^i_u - g) + \sum_{i \in \{3,\ldots,k\}, i \neq \ell_v} \bar{X}^i_u$. Therefore, $r_{\mathbf{z}} = \sum_{v \in H_{\mathbf{z}}} (1 - \bar{X}^{\ell_v}_v) = \sum_{v \in H_{\mathbf{z}}} \sum_{i \in \{1,\ldots,k\}, i \neq \ell_v} \bar{X}^i_v$ and so $r_z > |B'|g + \sum_{v \in H_{\mathbf{z}}} \Big[\sum_{i \in \{1,2\}, i \neq \ell_v} (\bar{X}^i_u - g) +$

$\sum_{i\in\{3,\dots,k\},i\neq\ell_v} \bar{X}_u^i\Big]$

$$
\begin{aligned}
r_z &> |B'|g + \sum_{i\in\{1,2\}} \sum_{v\in H_{\mathbf{z}},\ell_v\neq i} (\bar{X}_u^i - g) \\
&\quad + \sum_{i\in\{3,\dots,k\}} \sum_{v\in H_{\mathbf{z}},\ell_v\neq i} \bar{X}_u^i \\
&\geq \frac{k-2}{k-1}(r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r})g + \sum_{i\in\{1,2\}} (r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r})(\bar{X}_u^i - g) \\
&\quad + \sum_{i\in\{3,\dots,k\}} (r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r})\bar{X}_u^i \qquad (3) \\
&= (r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r})(1 - \frac{k}{k-1}g) = r_{\mathbf{z}},
\end{aligned}
$$

where inequality (3) uses the assumption that $|B_{-i}| \geq r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r}$ for all $i \in \{1,\dots,k\}$. The above calculations reaches a contradiction, $r_{\mathbf{z}} > r_{\mathbf{z}}$. Therefore, there exists an $i_{\mathbf{z}} \in \{1,\dots,k\}$ such that the number of nodes $v \in H_{\mathbf{z}}$ with $\ell_v \neq i_{\mathbf{z}}$ is at most $r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r}$, concluding the proof of the lemma. □

Proof of Theorem 1.3. Algorithm 1 sets $g = \frac{k-1}{k(r+1)}$. Firstly, note that by Lemma 3.2 the probability that an edge (u,v) is in the cut-set is at most $\frac{2k}{k-1}(r+1)d_{\bar{X}}(u,v)$, where $\bar{X}$ is an optimal solution to LP 1. Therefore, Algorithm 1 returns a cut-set with cut value at most $\frac{2k}{k-1}(r+1)\mathrm{OPT}_r$, where OPT_r denotes the optimal value of LP 1 with a move parameter of r.

Secondly, we prove that Algorithm 1 moves at most r nodes. To do this, we show that for each vector $\mathbf{z} \in A$, the number of nodes in $H_{\mathbf{z}}$ that are moved from their initial partitions is less than $r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r}$. There are two possibilities to consider here. Case (i): There exists a terminal in $s_i \in H_{\mathbf{z}}$ (recall that by Lemma 3.3, there can be at most one terminal in $H_{\mathbf{z}}$). In this case, Algorithm 1 assigns all the nodes in $H_{\mathbf{z}}$ to partition i. Therefore, the number of nodes that are moved to partitions different from their original partitions is equal to the number of nodes $v \in H_{\mathbf{z}}$ with $\ell_v \neq i$. By Lemma 3.4, the number of such nodes is less than $r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r}$. Case (ii): $H_{\mathbf{z}}$ has no terminal nodes. In this case, Algorithm 1 assigns all the nodes to partition $i^*_{\mathbf{z}} \leftarrow \arg\max_{j=1}^{k} |\{v \in H_{\mathbf{z}} | \ell_v = j\}|$. Defining B_i as the set of nodes $v \in H_{\mathbf{z}}$ with $\ell_v = i$, the number of nodes that are moved to partitions different from their original ones is equal to $|H_{\mathbf{z}}| - |B_{i^*_{\mathbf{z}}}|$. By Lemma 3.4, there exists a partition i for which $|H_{\mathbf{z}}| - |B_i| < r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r}$. By definition of $i^*_{\mathbf{z}}$, we have $|B_i| \leq |B_{i^*_{\mathbf{z}}}|$; therefore, $|H_{\mathbf{z}}| - |B_{i^*_{\mathbf{z}}}| < r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r}$.

The total number of nodes that are moved by Algorithm 1 in cases (i) and (ii) is strictly less than $\sum_{\mathbf{z}\in A} r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r}$. Using the definition of $r_{\mathbf{z}}$ and constraint (C7) of LP 1, we have $\sum_{\mathbf{z}\in A} r_{\mathbf{z}} + \frac{r_{\mathbf{z}}}{r} = (\frac{r+1}{r})\sum_{\mathbf{z}\in A} r_{\mathbf{z}} = (\frac{r+1}{r})\sum_{\mathbf{z}\in A}\sum_{v\in H_{\mathbf{z}}}(1 - \bar{X}_v^{\ell_v}) = (\frac{r+1}{r})\sum_{v\in V(G)}(1 - \bar{X}_v^{\ell_v}) \leq r+1$. This shows that the total number of nodes that Algorithm 1 moves is strictly less than $r+1$, in other words, at most r nodes are moved to partitions different from their original ones by the algorithm. □

3.2 Run Time and De-randomization

For computing the run time, note that finding the solution to LP1 takes $T_{LP}(n,m)$ time. The rest of the algorithm consists of simple loops and takes $O(mk)$ time.

One can de-randomize the algorithm using the technique of Calinescu, Karloff and Rabani Călinescu et al. (1998). They de-randomize their rounding for their $1.5 - 1/k$ approximation algorithm for the multiway-cut problem (at the expense of polynomial increase in running time), and they argue that there are $O(n)$ "interesting" values of ρ and hence one only needs to run the rounding algorithm for those values. To see this in our algorithm at a high level, note that for each i, there is a value ρ_i where for all $\rho < \rho_i$, $\tilde{X}_v^i$ is $g\lfloor\frac{\bar{X}_v^i}{g}\rfloor$ and for $\rho \geq \rho_i$, we have $\tilde{X}_v^i = g\lfloor\frac{\bar{X}_v^i}{g} + 1\rfloor$. The value of ρ_i can be computed from $\bar{X}_v^i$. So one can only run the algorithm for these values of ρ_i. For the running time, it increases by a factor of at most n, however one could decrease this factor by sorting the values of ρ.

4 EXPERIMENTS

We conduct a simple empirical assessment of our rounding algorithm (Algorithm 1) of Theorem 1.3. We remark that this experiment is *not* an extensive empirical evaluation of the algorithms, and the focus of it is on the results that are not well explained by theory, namely that the approximation factor of Algorithm 1 in practice is better than what Theorem 1.3 demonstrates. This suggests that there could be alternative analysis of our algorithm which results in better approximation guarantees for certain classes of graphs.

Set-up: We generate our sample graphs using the stochastic block model on 90 nodes as follows: First the 90 nodes are divided into three equally sized clusters. Then, between any two nodes in the same cluster we add an edge with probability p_H. Similarly, between any two nodes in different clusters we put an edge with probability p_L. We let $p_H = 0.3$ and $p_L = 0.1$. Note that this partitioning is very close to the optimal k-partitioning of the graph for $k = 3$; hence, we re-partition the constructed graph into three partitions uniformly at random. These new random partitions are then set as the initial partitioning of the graph. Following these steps, we make 100 random such graphs in total. As our benchmark, we consider the following simple greedy algorithm: The greedy algorithm has at most r rounds. At each round, the algorithm moves the node that decreases the value of the 3-cut by the largest amount. If at any point there is no such node, then the greedy algorithm

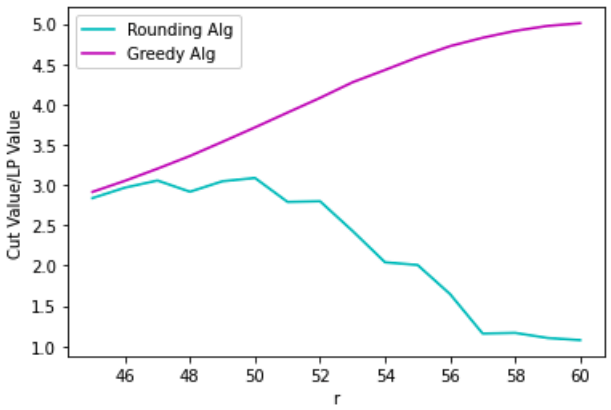

Figure 1: Performance of rounding (Algorithm 1) and greedy algorithms with respect to the solution of LP 1 for the explained stochastic block graphs.

halts. For each graph, we run Algorithm 1 for 30 different values of parameter ρ and take the 3-cut with the smallest value as the output.

Results: For each of the 100 random graph and each of the two algorithms (greedy and Algorithm 1), we compute the following ratios: The output of the algorithm divided by the objective value of LP 1. For each value of r in $[45, 60]$, we compute the average value of this ratio over all graphs, see Figure 1. For smaller values of r, we observe that both algorithms output similar cut values and they demonstrate similar performances. This could be due to the small size of our sample graphs and we believe the difference between the performances of these two algorithms is more evident with a larger sample size of graphs. While the greedy algorithm proves to perform reasonably well when the move parameter is bounded, we show in the full version of the paper (Behbahani et al., 2024) that our FPTAS algorithm can beat greedy in this case. As r approaches 60 and above, the LP solution is integer in most instances; hence, the Algorithm 1 does not play a significant role*.

5 CONCLUSION

This paper studies the r-move k-partitioning problem. We show that this problem is $W[1]$-hard and give simple and practical approximation algorithms for it. These algorithms are: an FPTAS for constant r, a $3(r+1)$-approximation algorithm and an $(O(1), O(1))$ bicriteria algorithm for general r. Our main results focus on LP rounding techniques. There remains several interesting open problems to understand the complexity of the r-move k-partitioning problem, some of them are listed below.

*One can see that if $\hat{r}$ is the number of nodes needed to be moved in order to get the optimal k-cut solution (in the absence of any budgetary constraints), then $\mathbb{E}[\hat{r}]$ is near 60.

(1) Is there an approximation algorithm for the r-move k-partitioning problem whose running time and approximation factor are independent of r? Recall that there exists an $O(\log(n))$-approximation algorithm for the Min r-size s-t cut problem using tree decomposition techniques. Could one generalize this algorithm to the r-move k-partitioning problem?

(2) Can one find any approximation lower bound for this problem? Note that similar partitioning problems, such as the MinSBCC problem (Hayrapetyan et al., 2005), do not have any approximation lower bounds yet.

References

Angelidakis, H., Makarychev, Y., and Manurangsi, P. (2017). An improved integrality gap for the călinescu-karloff-rabani relaxation for multiway cut. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 39–50. Springer.

Armon, A. and Zwick, U. (2006). Multicriteria global minimum cuts. *Algorithmica*, 46(1):15–26.

Behbahani, M., Dalirrooyfard, M., Fata, E., and Nevmyvaka, Y. (2024). Graph partitioning with limited moves. *arXiv preprint, arXiv:2402.15485*.

Bérczi, K., Chandrasekaran, K., Király, T., and Madan, V. (2020). Improving the integrality gap for multiway cut. *Mathematical Programming*, 183(1):171–193.

Bonnet, E., Escoffier, B., Paschos, V. T., and Tourniaire, E. (2015). Multi-parameter analysis for local graph partitioning problems: Using greediness for parameterization. *Algorithmica*, 71(3):566–580.

Cai, L. (2008). Parameterized complexity of cardinality constrained optimization problems. *The Computer Journal*, 51(1):102–121.

Călinescu, G., Karloff, H., and Rabani, Y. (1998). An improved approximation algorithm for multiway cut. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 48–52.

Chen, W., Samatova, N. F., Stallmann, M. F., Hendrix, W., and Ying, W. (2016). On size-constrained minimum s–t cut problems and size-constrained dense subgraph problems. *Theoretical Computer Science*, 609:434–442.

Chuzhoy, J., Makarychev, Y., Vijayaraghavan, A., and Zhou, Y. (2015). Approximation algorithms and hardness of the k-route cut problem. *ACM Transactions on Algorithms (TALG)*, 12(1):1–40.

Dahlhaus, E., Johnson, D. S., Papadimitriou, C. H., Seymour, P. D., and Yannakakis, M. (1992). The complexity of multiway cuts. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 241–251.

Dahlhaus, E., Johnson, D. S., Papadimitriou, C. H., Seymour, P. D., and Yannakakis, M. (1994). The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894.

Dickinson, S., Pelillo, M., and Zabih, R. (2001). Introduction to the special section on graph algorithms in computer vision. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 23(10):1049–1052.

Dinitz, M., Srinivasan, A., Tsepenekas, L., and Vullikanti, A. (2022). Fair disaster containment via graph-cut problems. In *International Conference on Artificial Intelligence and Statistics*, pages 6321–6333. PMLR.

Feige, U., Krauthgamer, R., and Nissim, K. (2003). On cutting a few vertices from a graph. *Discrete Applied Mathematics*, 127(3):643–649.

Feldmann, A. E. and Foschini, L. (2015). Balanced partitions of trees and applications. *Algorithmica*, 71(2):354–376.

Fomin, F. V., Golovach, P. A., and Korhonen, J. H. (2013). On the parameterized complexity of cutting a few vertices from a graph. In *International Symposium on Mathematical Foundations of Computer Science*, pages 421–432. Springer.

Goldschmidt, O. and Hochbaum, D. S. (1994). A polynomial algorithm for the k-cut problem for fixed k. *Mathematics of operations research*, 19(1):24–37.

Hayrapetyan, A., Kempe, D., Pál, M., and Svitkina, Z. (2005). Unbalanced graph cuts. In *European Symposium on Algorithms*, pages 191–202. Springer.

Hendrickson, B. and Kolda, T. G. (2000). Graph partitioning models for parallel computing. *Parallel computing*, 26(12):1519–1534.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 290–297.

Kahng, A. B., Lienig, J., Markov, I. L., and Hu, J. (2011). *VLSI physical design: from graph partitioning to timing closure*. Springer Science & Business Media.

Karger, D. R. and Stein, C. (1996). A new approach to the minimum cut problem. *Journal of the ACM (JACM)*, 43(4):601–640.

Khot, S. (2002). On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 767–775.

King, D. M., Jacobson, S. H., Sewell, E. C., and Cho, W. K. T. (2012). Geo-graphs: an efficient model for enforcing contiguity and hole constraints in planar graph partitioning. *Operations Research*, 60(5):1213–1228.

Lokshtanov, D. and Marx, D. (2013). Clustering with local restrictions. *Information and Computation*, 222:278–292.

Manokaran, R., Naor, J., Raghavendra, P., and Schwartz, R. (2008). Sdp gaps and ugc hardness for multiway cut, 0-extension, and metric labeling. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 11–20.

Räcke, H. (2008). Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 255–264.

Saran, H. and Vazirani, V. V. (1995). Finding k cuts within twice the optimal. *SIAM Journal on Computing*, 24(1):101–108.

Shah, N., Malensek, M., Shah, H., Pallickara, S., and Pallickara, S. L. (2019). Scalable network analytics for characterization of outbreak influence in voluminous epidemiology datasets. *Concurrency and Computation: Practice and Experience*, 31(7):e4998.

Sharma, A. and Vondrák, J. (2014). Multiway cut, pairwise realizable distributions, and descending thresholds. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 724–733.

Watanabe, T. and Nakamura, A. (1987). Edge-connectivity augmentation problems. *Journal of Computer and System Sciences*, 35(1):96–144.

Zhang, P. (2014). Unbalanced graph cuts with minimum capacity. *Frontiers of Computer Science*, 8:676–683.

Zhang, P. (2016). A new approximation algorithm for the unbalanced min s–t cut problem. *Theoretical Computer Science*, 609:658–665.

6 CHECKLIST

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No, we will do this upon the acceptance of our paper]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes, some are in the supplementary materials]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Not Applicable]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Not Applicable]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Not Applicable]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

SUPPLEMENTARY MATERIALS

7 HARDNESS RESULTS

In this section, we state our hardness results, including lower bounds on the integrality gap of LP 1 and our W[1]-hardness findings.

7.1 Integrality Gap of the r-Move k-Partitioning Linear Program

The following lemma discusses the integrality gap of the r-move k-partitioning linear program (i.e., LP 1).

Lemma 7.1 *LP 1 has an integrality gap of at least $r+1$.*

Proof. Let $P_1 : (v_1, \dots, v_{r+2})$ be a path in graph $G = (V, E)$ such that node v_1 is in partition 1 and the rest of the nodes on P_1 belong in partition 2, see Figure 2. Let the weight of the edge (v_1, v_2) be ϵ and the weight of all of the other edges on P_1 be one. Let $P_2 : (v_{r+3}, \dots, v_n)$ be a path with edges all of weight one. All nodes of P_2 belong in partition 2. Let v_1 and v_n be the terminals for partition 1 and 2, respectively. The defined partitions introduce a cut that has a weight of ϵ. Moving any node from P_2 to partition 1 only increases the cut value, so any optimal integer solution keeps all the nodes in P_2 in their initial partition. Moving any proper subset of nodes of P_1 to partition 1 increases the cut value to at least one. Furthermore, one cannot move all of the nodes in P_1 to partition 1 without violating the move constraint. So an optimal integer solution to LP 1 does not move any node in this graph and has a cut value of ϵ.

On the other hand, any (fractional) optimal solution of LP 1 on this graph has a cut value of at most $\frac{\epsilon}{r+1}$: For any node v_i, $i \in \{2, \dots, r+2\}$, this is achieved by setting $X^1_{v_i} = \frac{r}{r+1}$ and $X^2_{v_i} = \frac{1}{r+1}$. Let the nodes in P_2 stay in partition 2, i.e., $X^1_{v_i} = 0, X^2_{v_i} = 1$ for $i \in \{r+3, \dots, n\}$. The value of the cut in this solution is $\frac{\epsilon}{r+1}$. So the integrality gap is at least $r+1$. Note that this proof can be generalized to any $k > 2$ by adding dummy partitions with singletons in them. □

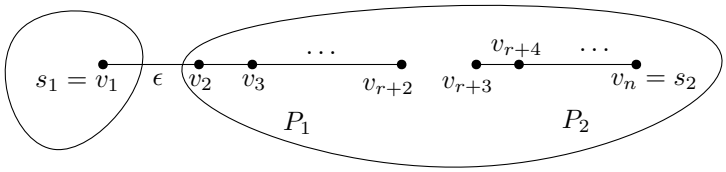

Figure 2: The example graph for the proof of Lemma 7.1.

Since the construction of Lemma 7.1 is in fact a Min r-size s-t cutconstruction, we have the following corollary.

Corollary 7.1 *The linear program of the Min r-size s-t cut problem stated in Zhang (2016) has an integrality gap of at least $r+1$.*

7.2 The r-Move k-Partitioning Problem is $W[1]$-hard

We begin by introducing two new notations here that will be useful for the proofs of this section and then discuss the computational complexity of the r-move k-partitioning problem. For a graph G and any subsets of its nodes $C_1, C_2 \subseteq V(G)$, let $E(C_1, C_2)$ refer to the set of edges between subsets C_1 and C_2. For any node $v \in C_1$, we define $\deg_{C_1}(v)$ as the number of neighbors of node v that are in C_1; that is, the number of nodes in C_1 that share an edge with node v.

Theorem 7.1 *Given an n-node graph G as an instance of the densest r-subgraph problem, there is an $O(n^2)$-node graph G' with initial partitions $\{A, B\}$ such that the densest r-subgraph of G has m^* edges if and only if the optimal solution of the r-move 2-partitioning problem on G' reduces the initial cut value by $2m^*$.*

Proof. We reduce the densest r-subgraph problem to the r-move 2-partitioning problem. Let G be an instance of the densest r-subgraph problem that has n nodes, m edges and unit edge weights. Using G, we construct a graph G' as an instance of the r-move 2-partitioning problem. Graph G' consists of two subgraphs A and B, where A is an exact copy of graph G (with n nodes and m edges) and B is a clique of size $2n^2 + n$. Furthermore, m nodes in B are reserved for the m edges of A; in other words, for each edge $e \in E(A)$ there exists a node $e \in V(B)$. Then, the endpoints of each edge $e = (u, v) \in E(A)$ gets connected to node $e \in V(B)$. Next, we add a terminal node t to subgraph A and connect it to all nodes in A. Similarly, a terminal node s is added to subgraph B and it gets connected to all nodes in $V(G')$. Note that after the addition of node s, subgraph B becomes a $2n^2 + n + 1$ clique. This finishes the construction of graph G', see Figure 3. Let the *initial cut* induced on G' split the graph into two partitions A and B. This cut has a value $2m + n < n^2 + 1$.

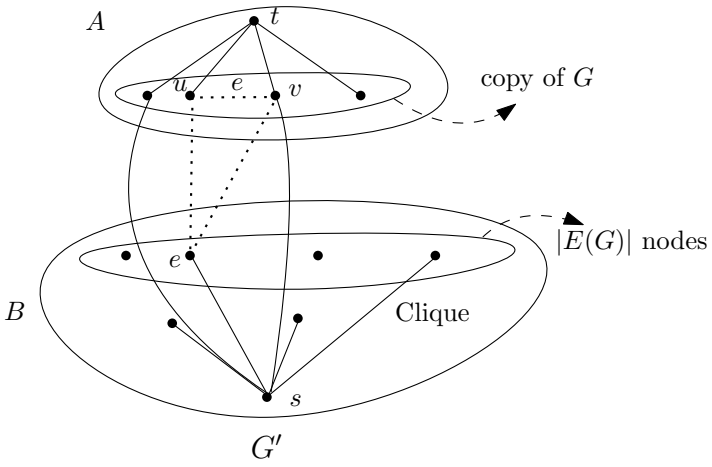

Figure 3: Reduction graph G' created from a densest r-subgraph instance G. The solid-line edges do not depend on G. For each edge $e = (u, v)$ in G, nodes u and v in A are connected to a node e in B.

First, we show that if we move a subset of nodes $X \subseteq V(A) \setminus \{t\}$ (along with all edges incident to the nodes in X) to subgraph B, then the cut value reduces by $2\big|E(X)\big|$. Note that, if we move a set of nodes $X \subseteq V(A) \setminus \{t\}$ to subgraph B, then the cut value changes by $\big|E\big(X, V(A) \setminus X\big)\big| - \big|E\big(X, V(B)\big)\big|$. Since subgraph induced by $V(A) \setminus \{t\}$ is exactly the same as graph G, we have

$$\begin{aligned}\big|E\big(X, V(A) \setminus X\big)\big| &= \big|E\big(X, \{t\}\big)\big| + \big|E\big(X, V(A) \setminus (X \cup \{t\})\big)\big| \\ &= \big|X\big| + \sum_{v \in X} \big[\deg_G(v) - \deg_X(v)\big].\end{aligned}$$

Next, we have

$$\begin{aligned}\big|E\big(X, V(B)\big)\big| &= \big|E\big(X, \{s\}\big)\big| + \big|E\big(X, V(B) \setminus \{s\}\big)\big| \\ &= \big|X\big| + \sum_{v \in X} \deg_G(v);\end{aligned}$$

thus, $\big|E\big(X, V(A) \setminus X\big)\big| - \big|E\big(X, V(B)\big)\big| = -\sum_{v \in X} \deg_X(v) = -2\big|E(X)\big|$.

Now, let X^* be the densest r-subgraph of G. If we move the copy of X^* in A from A to B, then the cut value reduces by $2m^* = 2\big|E(X^*)\big|$, as shown above. To prove the other direction, we show that if there exists a set of nodes $X \subseteq V(G')$ such that $|X| \leq r$ and moving each node in X to a partition different from the initial one it was assigned to reduces the cut by at least $2m^*$, then there exists an r-subgraph in G with m^* edges. To see this, first note that X cannot have any nodes from subgraph B. This is because the resulting cut would have a value at least $2n^2 + 1$ which is bigger than the value of the initial cut; consequently, $X \subseteq V(A)$. As shown earlier in this proof, by moving X from A to B the cut value is reduced by $2\big|E(X)\big|$, so $\big|E(X)\big| \geq m^*$. Therefore, the equivalent set of X in G has at least m^* edges, hence $m^* = \big|E(X)\big|$. Finally, the time it takes to build graph G' is $O\big(\big|V(G')\big| + \big|E(G')\big|\big) \subseteq O(n^4)$. □

Theorem 1.1 *The r-move k-partitioning problem with parameter r is W[1]-hard.*

Proof. For the r-move 2-partitioning problem, this comes from Theorem 7.1 and the W[1]-hardness of the densest r-subgraph problem (Cai, 2008). For the r-move k-partitioning problem with $k > 2$, by making the following modifications to graph G' we can make the same reduction as that of the proof of Theorem 7.1 work: adding $k-2$ dummy partitions to graph G', each containing one terminal that is not connected to any other node. Then, if Y is the set of nodes that are moved to one of these dummy partitions P, then by moving Y to one of the two main partitions the cut value will not increase. This is because Y has no edges to the terminal node in P. Therefore, all solutions consist of moving nodes to one of the two main partitions. □

Hardness of the Min r-size s-t cut problem: We now slightly change the above construction to fit to the Min r-size s-t cut problem.

Theorem 7.2 *Given an n-node graph G as an instance of the densest r-subgraph problem, there is an $O(n^2)$-node graph G' and a value $c(G')$ such that the densest r-subgraph of G has m^* edges if and only if the optimal solution of the Min r-size s-t cut problem on G' has a value of $c(G') - 2m^*$.*

Proof. Let A be a copy of the graph G. Add a terminal node t to A and connect it to all nodes in A with unit-weight

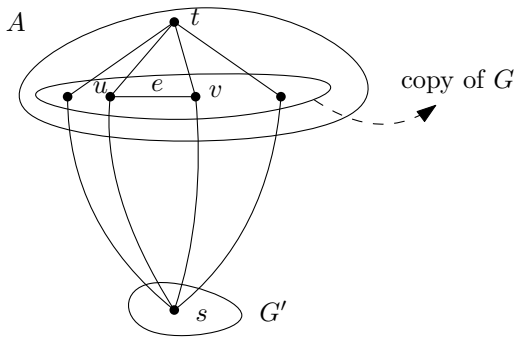

Figure 4: Reduction graph G' created from a densest r-subgraph instance G.

edges. Partition B consists of only one node, terminal s. For each $v \neq t$ in A, connect node v to terminal s with an edge of weight $1 + \deg_G(v)$, see Figure 4. Let $c(G')$ be the cut value of partitions $\{A, B\}$ which is equal to $2\big|E(G)\big| + \big|V(G)\big|$.

Here, we show that moving any subset of nodes $X \in V(A) \setminus \{t\}$ from partition A to B reduces the value of the cut induced by partitions $\{A, B)\}$ by $2\big|E(X)\big|$. To see this, note that after moving nodes X from A to B, the cut value is reduced by $\big|E\big(X, V(A) \setminus X\big)\big| - \big|E\big(X, V(B)\big)\big| = \big(|X| + \sum_{v\in X} \deg_G(v) - \deg_X(v)\big) - \big(\sum_{v\in X}(\deg_G(v) + 1)\big) = -2\big|E(X)\big|$. Suppose X^* is the densest r-subgraph in graph G and $m^* = \big|E(X^*)\big|$. Moving X^* to partition B reduces the cut by $2m^*$. If the optimal solution to the Min r-size s-t cut problem on G' is to move a set X from A to B, then this solution has a cut value of $c(G') - 2\big|E(X)\big|$, and so we must have that $\big|E(X)\big| \geq m^*$. Since $|X| \leq r$ and X^* is the densest r-subgraph in G, we must have $\big|E(X)\big| = m^*$. Note that if $|X| < r$, then we can add arbitrary nodes to X to make it have size r nodes without reducing its number of edges. □

From the above theorem and the W[1]-hardness of the densest r-subgraph problem we have that the weighted Min r-size s-t cut problem is W[1]-hard, thus, we have proved the following Corollary.

Corollary 7.2 *The Min r-size s-t cutproblem is W[1]-hard.*

Hardness of the r-move k-partitioning problem without terminals: In this paper, we mostly consider the r-move k-partitioning problem as a variant of the Multiway cut problem, i.e., we assume that the input graph has terminals that cannot be moved. However, one might wonder what the computational complexity of the r-move k-partitioning problem without terminals is. All our algorithmic results carry over to the r-move k-partitioning problem without terminals, since one can reduce the r-move k-partitioning problem without terminals to the r-move k-partitioning problem by adding dummy terminals for each partition as singletons. However, the main question in considering the r-move k-partitioning problem without terminals is whether it can be solved faster than the r-move k-partitioning problem, since in many partitioning problems the "with terminal" version of the problem is harder than the "without terminal" version. We show that our reduction works for the r-move k-partitioning problem without terminals as well, so this problem is also W[1]-hard. Thus, one cannot expect the complexity to change drastically by removing the terminals.

Theorem 7.3 *The r-move k-partitioning problem without terminals is W[1]-hard.*

Proof. Given a densest r-subgraph instance G, our reduction graph G' is the same as that of Theorem 7.1 except that we do not add terminal nodes s and t to the graph. The key observation here is that in the proof of Theorem 7.1 having terminal nodes s and t do not provide us with any specific benefit. We give a high level overview of the proof and the details can be easily derived from the proof of Theorem 7.1. We argue that moving a set of nodes X from partition A to B reduces the cut value by $2|E(X)|$. Then we can show that if the cut value is reduced by $2m^*$ after moving at most r nodes, then these nodes must all be in A. This is because the nodes in B create a clique and moving any of them to A increases the cut value. Moreover, the set of nodes that are moved must induce a densest r-subgraph in G in order for them to be an optimal solution to the r-move k-partitioning problem in G'. □