CUTS AND PARTITIONS: SOLVING, COUNTING, AND ENUMERATING

BY

CALVIN BEIDEMAN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2023

Urbana, Illinois

ABSTRACT

The problem of finding a global minimum cut in an undirected graph is fundamental to combinatorial optimization. It has numerous applications including network reliability, clustering, and the Travelling Salesman Problem. In addition to this computational problem, structural and enumerative aspects of minimum cuts are also foundational to representation and algorithmic results. The number of constant-approximate global minimum cuts in a connected graph is polynomial in the number of vertices. This structural result has applications in constructing cut sparsifiers, sketching and streaming algorithms, and approximation algorithms for TSP.

In this thesis we consider various generalizations of the minimum cut problem. We focus on solving these variants and on counting and enumerating optimum solutions. Our results include:

- A new and faster algorithm for computing connectivity in hypergraphs,
- The first deterministic polynomial time algorithm for enumerating hypergraph min-k-cut-sets,
- The first polynomial bound on the number of Multiobjective Min-cuts for a constant number of cost functions as well as the first polynomial time algorithm for enumerating all of them, and
- Inapproximability results for representing symmetric submodular functions using hypergraph cut functions.

ACKNOWLEDGMENTS

Firstly, I would like to thank my advisor, Karthik Chandrasekaran. I have been inspired by Karthik's diligence and enthusiasm for research. I have benefited from his willingness to go through many iterations of revision to make every paper and talk (and particularly this thesis) the best it can possibly be. Karthik also introduced me to the fascinating area of minimum cuts and partitions which became the focus of this thesis.

I would like to thank Chandra Chekuri, Sariel Har-Peled, and Sagnik Mukhopadhyay for serving on my thesis committee and for their valuable help and suggestions about this work. I am grateful to Chandra and Sagnik for their contributions to two of the papers on which this thesis is based, and to Sariel for engaging discussions on extensions of the work in this thesis which have led to further research. I also want to thank the others I have collaborated with in research, particularly Chao Xu, Weihang Wang, and Danupon Nanongkai, whose contributions to the work presented in this thesis were invaluable.

I also want to thank to all the professors I have had the chance to serve as a teaching assistant for. I have learned a lot from the teaching styles and methods of the professors at Illinois. Special thanks to Brad Solomon for co-teaching CS 173 with me in the summer, and for giving me enough support to make the experience much less stressful, while treating me as an equal in running the course.

I am very grateful for the entire theory group. I benefited both from discussions about our research, and from time spent sharing food or playing games. The group was a fun and welcoming place, and the camaraderie of my fellow students supported me through the challenges of grad school. Special thanks to Ian Ludden and Meghan Kelley for keeping me company in our office and to Qizheng He for our interesting discussions on min-cut in disk hypergraphs.

I would never have pursued a PhD without the influence of faculty from Carnegie Mellon. Special thanks to Anil Ada, Ryan O'Donnell, and Wes Pegden who taught courses that sparked my love for theory and graphs, and who each mentored me in research projects which prepared me for my doctoral studies. Anil also mentored me over my six semesters as a TA for 15-251 and his passion for teaching was a great inspiration to me to become a professor. I also want to Jeremiah Blocki, who gave me my very first taste of research back in high school, and who motivated me to study computer science at CMU.

Finally, I want to thank my parents, who raised me to love learning, and whose unfailing love and support carried me through the stress of my studies.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

The global min-cut problem in undirected graphs is central to combinatorial optimization with a rich history and many applications. A *cut* in an undirected graph is a partition of the vertices into two non-empty parts. The *value* of a cut is the sum of the costs of the edges with end-vertices in both parts. A min-cut is a cut with minimum value among all cuts in a graph. Closely related to the global min-cut problem is the min (s,t)-cut problem, where the input includes a (directed or undirected) graph as well as two special vertices s and t, and the goal is to find a cut separating s and t with minimum value. Min (s,t)-cuts are an important component in some of the results we will present. The min (s,t)-cut problem was studied as early as the 1950s, when Harris and Ross used it to model the problem of interdicting the soviet rail network, and Ford and Fulkerson gave a polynomial time algorithm for solving the problem by finding a maximum flow [1, 2, 3, 4]. The earliest algorithms for the global min-cut problem were based on algorithms for min (s,t)-cut. For a fixed vertex s, a min-cut must separate s from some other vertex t. Thus, computing the min (s,t)-cut between a fixed vertex s and every other vertex t and returning whichever of these cuts has the least value gives a global min-cut of an n-vertex graph in $n-1$ min (s,t)-cut computations [5, 6]. Min (s,t)-cuts have also been used in modern algorithms for global min-cut. In particular, Li and Panigrahi devised a minimum isolating cuts technique to show that the global min-cut problem can be solved deterministically using a polylogarithmic number of min (s,t)-cut computations [7]. The global min-cut problem has applications in numerous areas including network reliability [8], clustering [9], and image segmentation [10]. Min-cut algorithms are also used as subroutines in algorithms for other problems in graph theory and combinatorial optimization, such as the traveling salesman problem [11]. Because of these motivations, designing fast algorithms for solving global min-cut has been an active area of research [7, 12, 13, 14].

In addition to algorithms for finding a min-cut, research has also focused on understanding the structure of all min-cuts in a graph. In 1976, Dinitz, Karzanov, and Lomonosov [15] designed a concise representation of all min-cuts in a connected graph, known as the cactus representation. Using the cactus representation, they showed that the number of min-cuts in an n-vertex connected graph is at most $\binom{n}{2}$. Moreover, their proof leads to an algorithm to enumerate all min-cuts in a given graph in deterministic polynomial time. The upper bound of $\binom{n}{2}$ on the number of min-cuts in a connected graph is tight, as illustrated by the cycle-graph on n vertices. While [15] resolves counting and enumeration questions for min-cuts in graphs, subsequent works have considered generalizations of these counting and enumeration questions, such as counting the number of approximate min-cuts [16].

Combinatorial results on the number of min-cuts and the number of approximate min-cuts have been the crucial ingredients of several algorithmic and representation results in graphs. On the representation front, these counting results are foundational to algorithms for constructing compact representations of a graph, such as cut sparsifiers, which have applications in sketching and streaming [17, 18, 19, 20]. On the algorithmic front, these counting results allow for fast randomized construction of graph skeletons, which play a crucial role in fast algorithms to solve min-cut [21]. Furthermore, a polygon representation of the family of 6/5-approximate min-cuts in graphs was given by Benczur and Goemans [22, 23, 24]—this representation was used in the recent groundbreaking $(3/2-\epsilon)$-approximation for metric TSP [25].

Hypergraphs generalize graphs, and have applications in modern areas such as VLSI design and clustering [26, 27]. A hypergraph is specified by a set of vertices and a set of hyperedges, where each hyperedge is a subset of vertices. The *rank* of a hypergraph is the size of the largest hyperedge. Rank 2 hypergraphs correspond to graphs. Hyperedges can model relationships between more than two vertices and consequently, hypergraphs have more powerful modeling capability than graphs. This makes them useful for modeling structures such as social networks [28, 29] and satisfiability problems [30]. Algorithms for solving hypergraph min-cut have been known since the 1990s [31, 32, 33], and developing faster algorithms for this problem is an active research area, both for arbitrary rank hypergraphs [34, 35, 36] as well as for the special case where the rank is constant [20, 36]. Counting and enumeration results have also been extended to hypergraphs. In particular, the cactus representation of all min-cuts in a graph can be generalized to hypergraphs [37, 38], and leads to a deterministic polynomial time algorithm for enumerating all min-cut-sets in hypergraphs (the set of hyperedges intersecting both parts of a min-cut is called a min-cut-set).

Besides being of independent interest from both theoretical and practical perspectives, hypergraphs act as a natural bridge between graphs and symmetric submodular functions. A function $f\colon 2^V \to \mathbb{R}$, defined over a finite ground set V, is symmetric if $f(S) = f(V \setminus S)$ for every $S \subseteq V$. The function f is submodular if for every $A, B \subseteq V$, we have

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B). \tag{1.1}$$

Equivalently, f is submodular if for every $A, B \subseteq V$ with $A \subset B$ and every $v \in V \setminus B$ we have

$$f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B). \tag{1.2}$$

Inequality (1.2) represents the diminishing marginal returns property. Because of this diminishing marginal returns property, submodular functions arise naturally in economics [39] and game theory [40, 41], as well as in combinatorial optimization. Optimization over submodular functions also has applications in information theory [42], queuing theory [43], clustering [44], machine learning [45, 46], and computer vision [47]. An important example of a symmetric submodular function is the cut function of a graph/hypergraph. The cut function $d : 2^V \to \mathbb{R}$ of a graph $G = (V, E)$ with edge costs given by $c : E \to \mathbb{R}_{\geq 0}$ is defined as $d(X) := \sum_{e \in \delta(X)} c(e)$ where $\delta(X)$ is the set of edges with one end-vertex in X and one end-vertex in $V \setminus X$. Another important submodular function in combinatorial optimization is the matroid rank function. We note that the matroid rank function is not symmetric. A central problem related to submodular functions is submodular function minimization. In the submodular function minimization problem, we are given a finite set V and a submodular function $f : 2^V \to \mathbb{R}$ given via an oracle that returns $f(X)$ for every given $X \subseteq V$. The goal is to find an $X \subseteq V$ which minimizes $f(X)$. The ubiquity of submodular functions in combinatorial optimization has motivated the design of algorithms for submodular function minimization that are fast while also using few function evaluation queries [48, 49, 50, 51, 52, 53, 54, 55].

In this thesis we consider several generalizations of graph min-cut. We present both algorithmic as well as combinatorial results. For some problems we design faster algorithms, while for other problems we design the first polynomial time algorithm and the first deterministic polynomial time algorithm. We also use algorithms to understand the combinatorics of optimal solutions. All our algorithmic results are based on new structural properties of graphs and hypergraphs. In Chapter 2, we consider the generalization of the min-cut problem to hypergraphs. In Chapter 3, we consider the problem of counting the number of min-k-cuts in a hypergraph and of enumerating them. In Chapter 4, we consider the problem of counting and enumerating multicriteria variants of min-cut in graphs and hypergraphs. In Chapter 5, we consider the problem of representing arbitrary symmetric submodular functions using hypergraph cut functions. [1]

1.1 PRELIMINARIES

The set of positive integers less than or equal to n is denoted $[n]$. For functions $f(n)$ and $g(n)$ of n, we say that $f(n) = \tilde{O}(g(n))$ if $f(n) = O(g(n)\text{polylog}(n))$ and $f(n) = \hat{O}(g(n))$ if

[1]The research in this thesis was supported in part by NSF grants CCF-1814613 and CCF-1907937. I was also supported by the Saburo Muroga Endowed Fellowship and by the Computer Science Department at UIUC.

$f(n) = O(g(n)^{1+o(1)})$, where the $o(1)$ is with respect to n. We say that $f(n) = O_r(g(n))$ if $f(n) = O(g(n)h(r))$ for some function h.

1.1.1 Graphs and Hypergraphs

We define notation that is common between graphs and hypergraphs. For further details on concepts related to graphs, we refer the reader to standard textbooks (e.g. see [56]). A hypergraph $G = (V, E)$ is specified by a finite set V of vertices and a set E of hyperedges where each hyperedge $e \in E$ is a subset of V. The rank r of a hypergraph is defined as $\max_{e\in E} |e|$. If a hypergraph has rank 2, then it is a graph. We note that the space needed to represent a hypergraph is proportional to the sum of the sizes of all its hyperedges. Thus, we use $p := \sum_{e\in E} |e|$ to denote the size of the hypergraph. If the hypergraph is weighted, then it has an associated cost function $c\colon E \to \mathbb{R}_+$. We will assume throughout that edge costs are positive or at least non-negative—if negative costs are allowed, then solving min-cut becomes equivalent to solving the NP-hard max-cut problem. If all edges have cost 1, then we will refer to the hypergraph as unweighted or simple.

1.1.2 Cuts and Partitions

Let $G = (V, E)$ be a hypergraph. For every $\emptyset \subsetneq X \subsetneq V$ we call the 2-partition $\{X, V \setminus X\}$ a cut, and we call the set of hyperedges intersecting both parts of this partition a cut-set. For a subset $X \subseteq V$ of vertices, we define

$$\delta_G(X) := \{e \in E\colon e \cap X \neq \emptyset \text{ and } e \cap (V \setminus X) \neq \emptyset\}. \tag{1.3}$$

We say that $\delta_G(X)$ is the cut-set corresponding to the cut $\{X, V \setminus X\}$. The cut function $d_G\colon 2^V \to \mathbb{R}_{\geq 0}$ (with the subscript omitted when the hypergraph is clear from context) is defined by $d_G(X) := \sum_{e\in\delta_G(X)} c(e)$. The value of a cut $\{X, V \setminus X\}$ is $d_G(X)$. A min-cut is a cut with minimum value among all cuts, and a min-cut-set is the set of hyperedges intersecting both parts of a min-cut. We use $\lambda_G := \min_{\emptyset\subsetneq X\subsetneq V} d_G(X)$ to denote the min-cut value of G.

For $k \in [|V|]$, we call a partition $\{V_1, \ldots, V_k\}$ of V into k non-empty parts a k-cut. We define

$$\delta_G(V_1, \ldots, V_k) := \{e \in E\colon \exists i, j \in [k], i \neq j, e \cap V_i \neq \emptyset \text{ and } e \cap V_j \neq \emptyset\}. \tag{1.4}$$

We call $\delta_G(V_1, \ldots, V_k)$ a k-cut-set. The value of the k-cut $\{V_1, \ldots, V_k\}$ is $\sum_{e\in\delta_G(V_1,\ldots,V_k)} c(e)$. A min-$k$-cut is a cut with minimum value among all k-cuts. If $\{V_1, \ldots, V_k\}$ is a min-k-cut, then $\delta_G(V_1, \ldots, V_k)$ is called a min-k-cut-set.

1.2 THESIS CONTRIBUTIONS AND ORGANIZATION

The remainder of this thesis consists of four chapters, each of which is self-contained. We now discuss the contributions in each of these chapters.

1.2.1 Hypergraph Connectivity

In Chapter 2, we consider the problem of finding a min-cut in simple hypergraphs, which is also known as the hypergraph connectivity problem. In the hypergraph connectivity problem, we are given as input a simple (i.e. all hyperedges have unit cost, and there are no parallel hyperedges) hypergraph $G = (V, E)$ with n vertices and m edges, and the goal is to find a 2-partition of V such that the sum of the costs of hyperedges which cross this partition is minimized.

This problem is a generalization of the graph connectivity problem, which received recent attention after Kawarabayashi and Thorup [57] showed a structural theorem which led to a deterministic algorithm for graph connectivity that was faster than the known algorithms for weighted graphs. Several successive papers improved on this runtime for graph connectivity [58, 59]. Of particular relevance to our work is the simple almost linear time algorithm due to Saranurak, which was based on expander decomposition [60]. These results motivate the study of hypergraph connectivity, and the question of whether the structural theorem of Kawarabayashi and Thorup can be generalized to hypergraphs, or even constant rank hypergraphs. We show that, while the Kawarabayashi-Thorup structural theorem *does not* hold for hypergraphs, a weaker version of the theorem does hold and leads to a faster algorithm for hypergraph connectivity in low rank hypergraphs.

Our main result is the following theorem showing that hypergraph connectivity in constant rank hypergraphs can be solved faster than previously known.

Theorem 1.1. Let G be an r-rank n-vertex simple hypergraph of size p. Then, there exists a randomized algorithm that takes G as input and runs in time

$$\hat{O}_r\left(p + \min\left\{\lambda^{\frac{r-3}{r-1}}n^2, \frac{n^r}{\lambda^{\frac{r}{r-1}}}, \lambda^{\frac{5r-7}{4r-4}}n^{\frac{7}{4}}\right\}\right)$$

to return the connectivity λ of G with high probability. Moreover, the algorithm returns a min-cut in G with high probability.

Additionally, we give a deterministic version of our algorithm which improves on the runtime of the previous fastest known deterministic algorithm. Table 1.1 compares the run-times of our randomized and deterministic algorithms with the previous state of the art. We observe that our deterministic algorithm is faster than known algorithms when $r \geq 3$ is a constant and $\lambda = \Omega(n^{(r-2)/2})$, while our randomized algorithm is faster than known algorithms if $r \geq 3$ is a constant and $\lambda = n^{\Omega(1)}$. These results are based on joint work with Chandrasekaran, Mukhopadhyay, and Nanongkai and appeared in IPCO 2022 [61].

	Deterministic	**Randomized**
Previous run-time	$O(p + \lambda n^2)$ [38]	$\tilde{O}(p + \min\{\lambda n^2, n^r, \sqrt{pn(m+n)^{1.5}}\})$ [36, 38, 62]
Our run-time	$\hat{O}_r\left(p + \min\left\{\lambda n^2, \lambda^{\frac{r-3}{r-1}} n^2 + \frac{n^r}{\lambda}\right\}\right)$	$\hat{O}_r\left(p + \min\left\{\lambda^{\frac{r-3}{r-1}} n^2, \frac{n^r}{\lambda^{\frac{r}{r-1}}}, \lambda^{\frac{5r-7}{4r-4}} n^{\frac{7}{4}}\right\}\right)$

Table 1.1: Comparison of results to compute hypergraph connectivity (simple r-rank n-vertex m-hyperedge p-size hypergraphs with connectivity λ).

Subsequent to our work, Chen, Kyng, Liu, Peng, Gutenberg, and Sachdeva developed an almost linear time algorithm for max flow [63]. This algorithm, combined with the isolating cuts technique developed in [7] and generalized to hypergraphs by [64] and [62], gives an almost linear time randomized algorithm for hypergraph min-cut (even in weighted hypergraphs), which improves upon our result.

1.2.2 Enumerating Min-Cut-Sets and Min-k-Cut-Sets in Hypergraphs

In Chapter 3, we focus on the problem of *enumerating* min-cut-sets and min-k-cut-sets in hypergraphs. In the Hypergraph-k-Cut problem, we are given as input a hypergraph $G = (V, E)$ with n vertices and m edges, possibly with costs $c \colon E \to \mathbb{R}_+$ on the hyperedges, and the goal is to find a min-k-cut. We note that Hypergraph-MinCut is the case of Hypergraph-k-Cut where $k = 2$. Graph-k-Cut is the special case of Hypergraph-k-Cut in graphs. For constant $k \geq 2$, the number of min-k-cuts in an n-vertex connected graph is $O(n^k)$—this bound is tight and is a consequence of a recent improved analysis of a random contraction algorithm to solve Graph-k-Cut [65, 66, 67]; the same random contraction algorithm can also be used to enumerate all min-k-cuts in connected graphs in randomized polynomial time. Deterministic polynomial-time algorithms to enumerate all min-k-cuts in connected graphs are also known [68, 69, 70, 71]. While a polynomial time algorithm for Graph-k-Cut has been known for almost 30 years [72], the first polynomial time randomized [35] and deterministic [73] algorithms for Hypergraph-k-Cut for fixed constant k were developed only recently.

When discussing enumeration problems in hypergraphs, we focus on enumerating min-k-cut-*sets* rather than min-k-cuts. This is because, in contrast to graphs, where the number of min-k-cuts in a connected graph is polynomial for fixed k, the number of min-k-cuts in a connected hypergraph can be exponential. In fact, the number of min-2-cuts in a connected hypergraph can be exponential: Consider an n-vertex hypergraph which has a single hyperedge containing all of the vertices—we call this hypergraph the *spanning hyperedge example*. In the spanning hyperedge example, every cut is a min-cut, and thus the number of min-cuts is 2^{n-1}. Thus, we cannot hope to enumerate all min-k-cuts in a hypergraph. However, the spanning hyperedge example has only a single min-cut-set (the set containing the single hyperedge). It was recently shown that the number of min-k-cut-sets in a hypergraph is polynomial for every fixed constant k, and they can all be enumerated in randomized polynomial time [35]. In this thesis, we focus on deterministically enumerating min-k-cut-sets. Our main result is the following theorem.

Theorem 1.2. There is a deterministic polynomial-time algorithm for hypergraph min-k-cut-set enumeration for every fixed k.

The algorithm presented in this thesis is the first deterministic algorithm for hypergraph min-k-cut-set enumeration. It runs in time $pn^{O(k^2)}$. These results are based on joint work with Chandrasekaran and Wang and appeared in SODA 2022 [74]. Subsequently, in joint work with Chandrasekaran and Wang, we improved the deterministic runtime for hypergraph min-k-cut-set enumeration to $pn^{O(k)}$ [75].

1.2.3 Multicriteria Min-Cut

In Chapter 4, we consider multicriteria versions of the min-cut problem and show new counting and enumeration results in both graphs and constant rank hypergraphs. In multicriteria min-cut problems, instead of having a single cost function, each hyperedge has t different costs associated with it. For a hypergraph $G = (V, E)$, let $c_1, \ldots, c_t \colon E \to \mathbb{R}_{\geq 0}$ be the t different cost functions. There are several ways to extend the definition of a min-cut to this setting. We describe three of them in Section 4.1. Multicriteria variants have been studied for a wide variety of combinatorial optimization problems [76]. Many of these variants are intractable [77]. For example, in a natural multicriteria extension of the min (s, t)-cut problem we are given budgets $b_1, \ldots, b_{t-1}$ and asked to find an (s, t)-cut with minimum c_t-cost from among those which have c_i-cost at most b_i, for each $1 \leq i \leq t-1$.

This problem is NP-hard even for $t = 2$ criteria via a reduction from KNAPSACK. Similar hardness results can be shown for multicriteria generalizations of problems such as shortest path, minimum spanning tree, and maximum weight matching [77]. In contrast, certain multicriteria versions of the *global* min-cut problem are known to be solvable in polynomial time [78, 79].

For multicriteria optimization problems, the number of optimal solutions/points on the pareto-front is a commonly studied complexity measure and substantial work has focused on enumerating these solutions [77, 79, 80, 81, 82]. In Chapter 4 we focus on counting the number of optimal solutions to multicriteria min-cut problems in graphs and hypergraphs. We recall that the number of min-cuts in a hypergraph can be exponential, even with only a single cost function (recall the spanning hyperedge example). Hence, we focus on counting multicriteria *cut-sets* since we will be dealing mainly with hypergraphs. However, our results when specialized to graphs are equivalent to counting multicriteria cuts. We present several combinatorial and algorithmic results for different variants of multicriteria min-cut. We refer the reader to Chapter 4 for a description of the problems we consider and our results. These results are based on joint work with Chandrasekaran and Xu and appeared in Mathematical Programming [83].

1.2.4 Approximate Representation of Symmetric Submodular Functions using Hypergraph Cut Functions

In Chapter 5, we consider the problem of approximating symmetric submodular functions using hypergraph cut functions. For a parameter $\alpha \geq 1$, we say that a set function $g : 2^V \to \mathbb{R}_{\geq 0}$ α-approximates a set function $f : 2^V \to \mathbb{R}_{\geq 0}$ if

$$g(A) \leq f(A) \leq \alpha g(A) \ \forall \ A \subseteq V. \tag{1.5}$$

Given the prevalence of submodular functions in combinatorial optimization, a natural question that has been studied is whether an arbitrary submodular set function can be well-approximated by a concisely representable function. We distinguish between structural and algorithmic variants of this question: the structural question asks whether submodular functions can be well-approximated via concisely representable functions while the algorithmic question asks whether such a concise representation can be constructed using a polynomial number of function evaluation queries (note that the algorithmic question is concerned with the number of function evaluation queries as opposed to run-time). Concise representations with small-approximation factor are useful in learning, testing, streaming, and sketching algorithms. Consequently, concise representations with small-approximation factor for submodular functions (and their generalizations and subfamilies of submodular functions) have been studied from all these perspectives with most results focusing on monotone submodular functions [84, 85, 86, 87, 88, 89, 90, 91].

In this thesis, we focus on approximating *symmetric* submodular functions. Balcan, Harvey, and Iwata [85] showed that for every symmetric submodular function $f : 2^V \to \mathbb{R}_{\geq 0}$, there exists a function $g : 2^V \to \mathbb{R}_{\geq 0}$ defined by $g(S) := \sqrt{\chi(S)^T M \chi(S)}$, where $\chi(S) \in \{0,1\}^V$ is the indicator vector of $S \subseteq V$ and M is a symmetric positive definite matrix such that g $\sqrt{n}$-approximates f. We note that such a function g has a concise representation—namely, the matrix M. Is it possible to improve on the approximation factor for symmetric submodular functions using other concisely representable functions? The concisely representable family of functions that we study in this thesis is the family of hypergraph cut functions.

Our main result in Chapter 5 shows that arbitrary symmetric submodular functions cannot be well approximated by hypergraph cut functions. For a paramter $\alpha \geq 1$, we say that a symmetric submodular function $f : 2^V \to \mathbb{R}_{\geq 0}$ is α-hypergraph approximable if there exists a hypergraph on vertex set V with non-negative hyperedge costs whose cut function $d : 2^V \to \mathbb{R}_{\geq 0}$ α-approximates f. We show the following theorem.

Theorem 1.3. For every sufficiently large positive integer n, there exists a symmetric submodular function $f : 2^{[n]} \to \mathbb{Z}_{\geq 0}$ such that f is not α-hypergraph-approximable for

$$\alpha = o\left(\frac{n^{\frac{1}{3}}}{\log^2 n}\right).$$

We note that our proof of Theorem 1.3 is non-constructive. In addition to this main result, we also prove a positive result showing that certain symmetric submodular functions can be constant-factor approximated by hypergraph cut functions.

Theorem 1.4. Symmetrized rank functions of uniform matroids and partition matroids are 64-hypergraph-approximable.

There exist uniform matroids whose symmetrized rank functions cannot be $(n/4 - \epsilon)$-approximated by a graph cut function for all constant $\epsilon > 0$ [88]. Thus, Theorem 1.4 shows a strong separation between the representation power of graph and hypergraph cut functions. The results in Chapter 5 are based on joint work with Chandrasekaran, Chekuri, and Xu and appeared in FSTTCS 2022 [92].

CHAPTER 2: HYPERGRAPH CONNECTIVITY

In this chapter we design an algorithm for computing the connectivity of a hypergraph which runs in time $\hat{O}_r(p + \min\{\lambda^{\frac{r-3}{r-1}}n^2, n^r/\lambda^{\frac{r}{r-1}}, \lambda^{\frac{5r-7}{4r-4}}n^{\frac{7}{4}}\})$, where p is the size, n is the number of vertices, r is the rank (size of the largest hyperedge), and λ is the connectivity of the input hypergraph. Our algorithm also finds a minimum cut in the hypergraph. Our algorithm is faster than existing algorithms if $r = O(1)$ and $\lambda = n^{\Omega(1)}$. The heart of our algorithm is a structural result showing a trade-off between the number of hyperedges taking part in all minimum cuts and the minimum size of the smaller side of a minimum cut. This structural result can be viewed as a generalization of a well-known structural theorem for simple graphs [57]. We extend the framework of expander decomposition to hypergraphs to prove this structural result. In addition to the expander decomposition framework, our faster algorithm also relies on a new near-linear time procedure to compute connectivity when one of the sides in a minimum cut is small. The results in this chapter are based on joint work with Chandrasekaran, Mukhopadhyay, and Nanongkai and appeared in IPCO '22 [61]. The research in this chapter was supported in part by NSF grants CCF-1814613 and CCF-1907937.

2.1 INTRODUCTION

A hypergraph $G = (V, E)$ is specified by a vertex set V and a collection E of hyperedges, where each hyperedge $e \in E$ is a subset of vertices. In this chapter, we address the problem of computing connectivity/global min-cut in hypergraphs with low rank (e.g., constant rank). The *rank* of a hypergraph, denoted r, is the size of the largest hyperedge—in particular, if the rank of a hypergraph is 2, then the hypergraph is a graph. In the global min-cut problem, the input is a hypergraph with hyperedge weights $w : E \to \mathbb{R}_+$, and the goal is to find a minimum weight subset of hyperedges whose removal disconnects the hypergraph. Equivalently, the goal is to find a partition of the vertex set V into two non-empty parts $(C, V \setminus C)$ so as to minimize the weight of the set of hyperedges intersecting both parts. For a subset $C \subseteq V$, we will denote the weight of the set of hyperedges intersecting both C and $V \setminus C$ by $d(C)$, the resulting function $d : V \to \mathbb{R}_+$ as the cut function of the hypergraph, and the weight of a min-cut by $\lambda(G)$ (we will use λ when the graph G is clear from context).

If the input hypergraph is *simple*—i.e., each hyperedge has unit weight and no parallel copies—then the weight of a min-cut is also known as the *connectivity* of the hypergraph. We focus on finding connectivity in hypergraphs. We emphasize that, in contrast to graphs whose representation size is the number of edges, the representation size of a hypergraph $G = (V, E)$ is $p := \sum_{e \in E} |e|$. We note that $p \le rm$, where r is the rank and m is the number of hyperedges in the hypergraph, and moreover, $r \le n$, where n is the number of vertices. We emphasize that the number of hyperedges m in a hypergraph could be exponential in the number of vertices.

Previous work. Since the focus of our work is on simple unweighted hypergraphs, we discuss previous work for computing global min-cut in simple unweighted hypergraphs (i.e., computing connectivity) here (see Section 2.1.3 for a discussion of previous works on computing global min-cut in weighted hypergraphs/graphs). The current fastest algorithms to compute graph connectivity (i.e., when $r = 2$) are randomized and run in time $\tilde{O}(m)$ [12, 13, 21, 57, 58, 59]. In contrast, algorithms to compute hypergraph connectivity are much slower. Furthermore, for hypergraph connectivity/global min-cut, the known randomized approaches are not always faster than the known deterministic approaches. There are two broad algorithmic approaches for global min-cut in hypergraphs: vertex-ordering and random contraction. We discuss these approaches now.

Nagamochi and Ibaraki [93] introduced a groundbreaking vertex-ordering approach to solve global min-cut in graphs in time $O(mn)$. In independent works, Klimmek and Wagner [31] as well as Mak and Wong [32] gave two different generalizations of the vertex-ordering approach to compute hypergraph connectivity in time $O(pn)$. Queyranne [33] generalized the vertex-ordering approach further to solve *non-trivial symmetric submodular minimization*.[2] Queyranne's algorithm can be implemented to compute hypergraph connectivity in time $O(pn)$. Thus, all three vertex-ordering based approaches to compute hypergraph connectivity have a run-time of $O(pn)$. This run-time was improved to $O(p + \lambda n^2)$ by Chekuri and Xu [38]: They designed an $O(p)$-time algorithm to construct a *min-cut-sparsifier*, namely a subhypergraph G' of the given hypergraph with size $p' = O(\lambda n)$ such that $\lambda(G') = \lambda(G)$. Applying the vertex-ordering based algorithm to G' gives the connectivity of G within a run-time of $O(p + \lambda n^2)$.

We emphasize that all algorithms discussed in the preceding paragraph are deterministic. Karger [16] introduced the influential random contraction approach to solve global min-cut in graphs which was adapted by Karger and Stein [65] to design an $O(n^2 \log^3)$ time algorithm. Kogan and Krauthgamer [20] extended the random contraction approach to solve global min-

[2]The input here is a symmetric submodular function $f : 2^V \to \mathbb{R}$ via an evaluation oracle and the goal is to find a partition of V into two non-empty parts $(C, V \setminus C)$ to minimize $f(C)$. We recall that a function $f : 2^V \to \mathbb{R}$ is symmetric if $f(A) = f(V \setminus A)$ for all $A \subseteq V$ and is submodular if $f(A) + f(B) \ge f(A \cap B) + f(A \cup B)$ for all $A, B \subseteq V$. The cut function of a hypergraph $d : V \to \mathbb{R}_+$ is symmetric and submodular.

cut in r-rank hypergraphs in time $\tilde{O}_r(mn^2)$ [3]. Ghaffari, Karger, and Panigrahi [34] suggested a non-uniform distribution for random contraction in hypergraphs and used it to design an algorithm to compute hypergraph connectivity in $\tilde{O}((m+\lambda n)n^2)$ time. Chandrasekaran, Xu, and Yu [35] refined their non-uniform distribution to obtain an $O(pn^3 \log n)$ time algorithm for global min-cut in hypergraphs. Fox, Panigrahi, and Zhang [36] proposed a branching approach to exploit the refined distribution leading to an $O(p + n^r \log^2 n)$ time algorithm for hypergraph global min-cut, where r is the rank of the input hypergraph. Chekuri and Quanrud [62] designed an algorithm based on isolating cuts which achieves a runtime of $\tilde{O}(\sqrt{pn(m+n)^{1.5}})$ for global min-cut in hypergraphs.

Thus, prior to our work, the fastest known algorithm to compute hypergraph connectivity was a combination of the algorithms of Chekuri and Xu [38], Fox, Panigrahi, and Zhang [36], and Chekuri and Quanrud [62] with a run-time of

$$\tilde{O}\left(p + \min\left\{\lambda n^2, n^r, \sqrt{pn(m+n)^{1.5}}\right\}\right). \tag{2.1}$$

2.1.1 Our Results

In this chapter, we improve the run-time to compute hypergraph connectivity in low rank simple hypergraphs.

Theorem 2.1. [Algorithm] Let G be an r-rank n-vertex simple hypergraph of size p. Then, there exists a randomized algorithm that takes G as input and runs in time

$$\hat{O}_r\left(p + \min\left\{\lambda^{\frac{r-3}{r-1}} n^2, \frac{n^r}{\lambda^{\frac{r}{r-1}}}, \lambda^{\frac{5r-7}{4r-4}} n^{\frac{7}{4}}\right\}\right)$$

to return the connectivity λ of G with high probability. Moreover, the algorithm returns a min-cut in G with high probability.

See Section 2.4.5 for the run-time of our algorithm in the O-notation. Our techniques can also be used to obtain a deterministic algorithm that runs in time

$$\hat{O}_r\left(p + \min\left\{\lambda n^2, \lambda^{\frac{r-3}{r-1}} n^2 + \frac{n^r}{\lambda}\right\}\right). \tag{2.2}$$

Our deterministic algorithm is faster than Chekuri and Xu's algorithm when r is a constant and $\lambda = \Omega(n^{(r-2)/2})$, while our randomized algorithm is faster than known algorithms if r is

[3]For functions $f(n)$ and $g(n)$ of n, we say that $f(n) = \tilde{O}(g(n))$ if $f(n) = O(g(n)\text{polylog}(n))$ and $f(n) = \hat{O}(g(n))$ if $f(n) = O(g(n)^{1+o(1)})$, where the $o(1)$ is with respect to n. We say that $f(n) = O_r(g(n))$ if $f(n) = O(g(n)h(r))$ for some function h. We define $\tilde{O}_r(f(n))$ and $\hat{O}_r(f(n))$ analogously.

a constant and $\lambda = n^{\Omega(1)}$. We summarize the previous fastest algorithms and our results in Table 2.1.

	Deterministic	**Randomized**
Previous run-time	$O(p + \lambda n^2)$ [38]	$\tilde{O}(p + \min\{\lambda n^2, n^r, \sqrt{pn(m+n)^{1.5}}\})$ [36, 38, 62]
Our run-time	$\hat{O}_r\left(p + \min\left\{\lambda n^2, \lambda^{\frac{r-3}{r-1}} n^2 + \frac{n^r}{\lambda}\right\}\right)$	$\hat{O}_r\left(p + \min\left\{\lambda^{\frac{r-3}{r-1}} n^2, \frac{n^r}{\lambda^{\frac{r}{r-1}}}, \lambda^{\frac{5r-7}{4r-4}} n^{\frac{7}{4}}\right\}\right)$

Table 2.1: Comparison of results to compute hypergraph connectivity (simple unweighted r-rank n-vertex m-hyperedge p-size hypergraphs with connectivity λ).

Our algorithm for Theorem 2.1 proceeds by considering two cases: either (i) the hypergraph has a min-cut where one of the sides is small or (ii) both sides of every min-cut in the hypergraph are large. To account for case (i), we design a near-linear time algorithm to compute a min-cut; to account for case (ii), we perform contractions to reduce the size of the hypergraph without destroying a min-cut and then run known algorithms on the smaller-sized hypergraph leading to savings in run-time. Our contributions in this chapter are twofold: (1) On the algorithmic front, we design a near-linear time algorithm to find a min-cut where one of the sides is small (if it exists); (2) On the structural front, we show a trade-off between the number of hyperedges taking part in all minimum cuts and the minimum size of the smaller side of a minimum cut (see Theorem 2.2). This structural result is a generalization of the acclaimed Kawarabayashi-Thorup graph structural theorem [57, 94] (Fulkerson prize 2021). We use the structural result to reduce the size of the hypergraph in case (ii). We elaborate on this structural result now.

Theorem 2.2. [Structure] Let $G = (V, E)$ be an r-rank n-vertex simple hypergraph with m hyperedges and connectivity λ. Suppose $\lambda \geq r(4r^2)^r$. Then, at least one of the following holds:

1. There exists a min-cut $(C, V \setminus C)$ such that
$$\min\{|C|, |V \setminus C|\} \leq r - \frac{\log\left(\frac{\lambda}{4r}\right)}{\log n},$$
2. The number of hyperedges in the union of all min-cuts is
$$O\left(r^{9r^2+2}\left(\frac{6r^2}{\lambda}\right)^{\frac{1}{r-1}} m \log n\right) = \tilde{O}_r\left(\frac{m}{\lambda^{\frac{1}{r-1}}}\right).$$

The Kawarabayashi-Thorup structural theorem for graphs [57, 94] states that if every min-cut is non-trivial, then the number of edges in the union of all min-cuts is $O(m/\lambda)$, where a cut is defined to be *non-trivial* if it has at least two vertices on each side. Substituting $r = 2$ in our structural theorem recovers this known Kawarabayashi-Thorup structural theorem for graphs. We emphasize that the Kawarabayashi-Thorup structural theorem for graphs is the backbone of the current fastest algorithms for computing connectivity in graphs and has been proved in the literature via several different techniques [57, 58, 59, 60, 95]. Part of the motivation behind our work was to understand whether the Kawarabayashi-Thorup structural theorem for graphs could hold for constant rank hypergraphs and if not, then what would be an appropriate generalization. We discovered that the Kawarabayashi-Thorup graph structural theorem *does not* hold for hypergraphs: There exist hypergraphs in which (i) the min-cut capacity λ is $\Omega(n)$, (ii) there are no trivial min-cuts, and (iii) the number of hyperedges in the union of all min-cuts is a constant fraction of the number of hyperedges—see Lemma 2.11 in Section 2.7 for such an example. The existence of such examples suggests that we need an alternative definition of *trivial min-cuts* if we hope to extend the Kawarabayashi-Thorup structural theorem for graphs to r-rank hypergraphs. Conclusion 1 of Theorem 2.2 can be viewed as a way to redefine the notion of *trivial min-cuts*. We denote the *size* of a cut $(C, V \setminus C)$ to be $\min\{|C|, |V \setminus C|\}$—we emphasize that the size of a cut refers to the size of the smaller side of the cut as opposed to the capacity of the cut. A min-cut is *small-sized* if the smaller side of the cut has at most $r - \log(\lambda/4r)/\log n$ many vertices. With this definition, Conclusion 2 of Theorem 2.2 can be viewed as a generalization of the Kawarabayashi-Thorup structural theorem to hypergraphs which have no small-sized min-cuts: it says that if no min-cut is small-sized, then the number of hyperedges in the union of all min-cuts is $\tilde{O}_r(m/\lambda^{\frac{1}{r-1}})$.

We mention that the factor $\lambda^{-1/(r-1)}$ in Conclusion 2 of Theorem 2.2 cannot be improved: There exist hypergraphs in which every min-cut has at least $\sqrt{n}$ vertices on both sides and the number of hyperedges in the union of all min-cuts is $\Theta(m \cdot \lambda^{-1/(r-1)})$—see Lemma 2.10 in Section 2.6. We also note that the structural theorem holds only for *simple* hypergraphs/graphs and is known to fail for weighted graphs. As a consequence, our algorithmic techniques are applicable only in simple hypergraphs and not in weighted hypergraphs.

Subsequent to our work, Chen, Kyng, Liu, Peng, Gutenberg, and Sachdeva [63] developed an almost-linear time algorithm for min-cost max-flow. Min (s, t)-cut in hypergraphs can be solved by computing a maximum flow in the bipartite representation of the hypergraph, which has $m + n$ vertices and p edges. Thus, the algorithm of [63] gives an $O(p^{1+o(1)})$ algorithm for hypergraph min (s, t)-cut. Furthermore, the isolating cut approach of Li and Panigrahi [7] gives a randomized algorithm for solving Hypergraph-MinCut using a polylogarithmic number of hypergraph min (s,t)-cut computations [62, 64]. Thus, the algorithm of [63] gives a randomized algorithm for HYPERGRAPH-MINCUT in arbitrary rank hypergraphs with a runtime of $O(p^{1+o(1)})$. This algorithm is asymptotically faster than our algorithm for all values of r and λ.

2.1.2 Technical Overview

Concepts used in the proof strategy of Theorem 2.2 will be used in the algorithm of Theorem 2.1 as well, so it will be helpful to discuss the proof strategy of Theorem 2.2 before the algorithm. We discuss this now. We define a cut $(C, V \setminus C)$ to be *moderate-sized* if $\min\{|C|, |V\setminus C|\} \in (r-\log(\lambda/4r)/\log n, 4r^2)$ and to be *large-sized* if $\min\{|C|, |V\setminus C|\} \geq 4r^2$; we recall that the cut $(C, V \setminus C)$ is small-sized if $\min\{|C|, |V \setminus C|\} \leq r - \log(\lambda/4r)/\log n$.

Proof strategy for the structural theorem (Theorem 2.2). We assume that $\lambda > r(4r^2)^r$ as in the statement of Theorem 2.2. The first step of our proof is to show that every min-cut in a hypergraph is either large-sized or small-sized but not moderate-sized—in particular, we prove that if $(C, V \setminus C)$ is a min-cut with $\min\{|C|, |V \setminus C|\} < 4r^2$, then it is in fact a small-sized min-cut (Lemma 2.1 with the additional assumption that $\lambda > r(4r^2)^r$). Here is the informal argument: For simplicity, we will show that if $(C, V \setminus C)$ is a min-cut with $\min\{|C|, |V \setminus C|\} < 4r^2$, then $\min\{|C|, |V \setminus C|\} \leq r$. For the sake of contradiction, suppose that $\min\{|C|, |V \setminus C|\} > r$. The crucial observation is that since the hypergraph has rank r, no hyperedge can contain the smaller side of the min-cut entirely. The absence of such hyperedges means that even if we pack hyperedges in G as densely as possible while keeping $(C, V \setminus C)$ as a min-cut, we cannot pack sufficiently large number of hyperedges to ensure that the degree of each vertex is at least λ. A more careful counting argument extends this proof approach to show that $\min\{|C|, |V \setminus C|\} \leq r - \log\lambda/\log n$.

Now, in order to prove Theorem 2.2, it suffices to prove Conclusion 2 under the assumption that all min-cuts are large-sized, i.e., $\min\{|C|, |V \setminus C|\} \geq 4r^2$ for every min-cut $(C, V \setminus C)$. Our strategy to prove Conclusion 2 is to find a partition of the vertex set V such that (i) every hyperedge that is completely contained in one of the parts does not cross any min-cut, and (ii) the number of hyperedges that intersect multiple parts (and therefore, possibly cross some min-cut) is small, i.e., $\tilde{O}_r(m\cdot\lambda^{-1/(r-1)})$. To this end, we start by partitioning the vertex set of the hypergraph G into $X_1, \ldots, X_k$ such that the total number of hyperedges intersecting more than one part of the partition is $\tilde{O}_r(m \cdot \lambda^{-1/(r-1)})$ and the subhypergraph induced by each X_i has conductance $\Omega_r(\lambda^{-1/(r-1)})$ (see Section 2.2 for the definition of conductance)—such a decomposition is known as an *expander decomposition*. An expander decomposition immediately satisfies (ii) since the number of hyperedges intersecting more than one part is small. Unfortunately, it may not satisfy (i); yet, it is very close to satisfying (i)—we can guarantee that for every min-cut $(C, V \setminus C)$ and every X_i, either C includes very few vertices from X_i, or C includes almost all the vertices of X_i i.e., $\min\{|X_i \cap C|, |X_i \setminus C|\} = O_r(\lambda^{1/(r-1)})$. We note that if $\min\{|X_i \cap C|, |X_i \setminus C|\} = 0$ for every min-cut $(C, V \setminus C)$ and every part X_i then (i) would be satisfied; moreover, if a part X_j is a singleton vertex part (i.e., $|X_j| = 1$), then $\min\{|X_j \cap C|, |X_j \setminus C|\} = 0$ holds. So, our strategy, at this point, is to remove some of the vertices from X_i to form their own singleton vertex parts in the partition in order to achieve $\min\{|X_i \cap C|, |X_i \setminus C|\} = 0$ while controlling the increase in the number of hyperedges that cross the parts. This is achieved by a TRIM operation and a series of SHAVE operations.

The crucial parameter underlying TRIM and SHAVE operations is the notion of degree within a subset: We will denote the degree of a vertex v as $d(v)$ and define the degree contribution of a vertex v inside a vertex set X, denoted by $d_X(v)$, to be the number of hyperedges containing v that are completely contained in X. The TRIM operation on a part X_i repeatedly removes from X_i vertices with small degree contribution inside X_i, i.e. $d_{X_i}(v) < d(v)/2r$, until no such vertex can be found. Let X_i' denote the set obtained from X_i after the TRIM operation. We note that our partition now consists of $X_1', \ldots, X_k'$ as well as singleton vertex parts for each vertex that we removed with the TRIM operation. This operation alone makes a lot of progress towards our goal—we show that $\min\{|X_i' \cap C|, |X_i' \setminus C|\} = O(r^2)$, while the number of hyperedges crossing the partition blows up only by an $O(r)$ factor (see Claims 2.1 and 2.3). The little progress that is left to our final goal is achieved by a series of ($O(r^2)$ many) SHAVE operations. The SHAVE operation finds the set of vertices in each X_i' whose degree contribution inside X_i' is not very large, i.e., $d_{X_i'}(v) \leq (1 - r^{-2})d(v)$ and removes this set of vertices from X_i' in one shot—such vertices are again declared as singleton vertex parts in the partition. We show that the SHAVE operation strictly reduces $\min\{|X_i' \cap C|, |X_i' \setminus C|\}$ without adding too many hyperedges across the parts (see Claims 2.2 and 2.4)—this argument crucially uses the assumption that all min-cuts are large-sized (i.e., $\min\{|C|, |V \setminus C|\} \geq 4r^2$). Because of our guarantee from the TRIM operation regarding $\min\{|X_i' \cap C|, |X_i' \setminus C|\}$, we need to perform the SHAVE operation $O(r^2)$ times to obtain a partition that satisfies conditions (i) and (ii) stated in the preceding paragraph.

Algorithm from structural theorem (Theorem 2.1). We now describe how to use the structural theorem (Theorem 2.2) to design an algorithm for hypergraph connectivity with run-time guarantee as stated in Theorem 2.1. If the min-cut capacity λ is small, then existing algorithms are already fast, so we assume that the min-cut capacity λ is sufficiently large. With this assumption, we consider the two conclusions mentioned in Theorem 2.2.

We design a near-linear time algorithm for finding small-sized min-cuts if the first conclusion holds. If the first conclusion fails, then the second conclusion holds; now, in order to obtain an algorithm for min-cut, we make our proof of Theorem 2.2 constructive—we obtain a small-sized hypergraph which retains all hyperedges that participate in min-cuts; we run existing min-cut algorithms on this small-sized hypergraph. We note here that, in a recent result, Saranurak [60] used a constructive version of a similar theorem to design an algorithm for computing connectivity in simple graphs in almost linear time (with a run-time of $m^{1+o(1)}$). Since our structural theorem is meant for hypergraph connectivity (and is hence, more complicated than what is used by [60]), we have to work more.

We now briefly describe our algorithm: Given an r-rank hypergraph G, we estimate the connectivity λ to within a constant factor in $O(p)$ time using an algorithm of Chekuri and Xu [38]. Next, we run the sparsification algorithm of [38] in time $O(p)$ to obtain a new hypergraph G' with size $p' = O(\lambda n)$ such that all min-cuts are preserved. The rest of the steps are run on this new hypergraph G'. We have two possibilities as stated in Theorem 2.2. We account for these two possibilities by running two different algorithms: (i) Assuming that some min-cut has size less than $r - \log(\lambda/4r)/\log n$, we design a near-linear time algorithm to find a min-cut (see Theorem 2.10). This algorithm is inspired by recent vertex connectivity algorithms, in particular the local vertex connectivity algorithm of [96, 97] and the sublinear-time kernelization technique of [98]. This algorithm runs in $\tilde{O}_r(p)$ time. (ii) Assuming that every min-cut is large-sized, we design a fast algorithm to find a min-cut (see Theorem 2.14). For this, we find an expander decomposition $\mathcal{X}$ of G', perform a TRIM operation followed by a series of $O(r^2)$ SHAVE operations, and then contract each part of the trimmed and shaved expander decomposition to obtain a hypergraph G''. This reduces the number of vertices in G'' to $O_r(n/\lambda^{1/(r-1)})$ and consequently, running the global min-cut algorithm of either [36] or [38] or [62] (whichever is faster) on G'' leads to an overall run-time of $\hat{O}_r(p + \min\{\lambda^{(r-3)/(r-1)}n^2, n^r/\lambda^{r/(r-1)}, \lambda^{(5r-7)/(4r-4)}n^{7/4}\})$ for step (ii). We return the cheaper of the two cuts found in steps (i) and (ii). The correctness of the algorithm follows by the structural theorem and the total run-time is $\hat{O}_r(p + \min\{n^r/\lambda^{r/(r-1)}, \lambda^{(r-3)/(r-1)}n^2, \lambda^{(5r-7)/(4r-4)}n^{7/4}\})$.

We note here that the expander decomposition framework for graphs was developed in a series of works for the dynamic connectivity problem [99, 100, 101, 102]. Very recently, it has found applications for other problems [103, 104, 105]. Closer to our application, Saranurak [60] used expander decomposition to give an algorithm to compute edge connectivity in graphs via the use of TRIM and SHAVE operations. The TRIM and SHAVE operations were introduced by Kawarabayashi and Thorup [57] to compute graph connectivity in deterministic $O(m \log^{12} n)$ time. Our line of attack is an adaptation of Saranurak's approach.

Organization. In Section 2.1.3, we discuss algorithms for the more general problem of weighted hypergraph min-cut. In Section 2.2, we introduce the necessary preliminaries. In Section 2.3, we prove our structural theorem (Theorem 2.2). In Section 2.4, we present our hypergraph min-cut algorithm and prove Theorem 2.1. In Section 2.5, we give a deterministic variant of our min-cut algorithm from Section 2.4.1 for small-sized min-cut—this is necessary for the (slightly slower) deterministic version of our min-cut algorithm. In Section 2.6, we provide a tight example for our structural theorem (i.e., Theorem 2.2). In Section 2.7, we provide an example to justify the modified notion of non-triviality while considering min-cuts in hypergraphs. In Section 2.8 we conclude by discussing further work on hypergraph connectivity and some related open problems. For the technical sections, we encourage the reader to consider $r = O(1)$ during first read.

2.1.3 Relevant Work

We briefly describe the approaches known for global min-cut in weighted graphs and hypergraphs. We begin by discussing the case of graphs. On the deterministic front, until recently, the fastest known algorithm for global min-cut was $O(mn)$ due to Nagamochi and Ibaraki [93]. This was recently improved to $\tilde{O}(m\cdot\min\{\sqrt{m}, n^{2/3}\})$ by Li and Panigrahi [7] and then to $O(m^{1+o(1)})$ by Li [14]. We now discuss the randomized approaches. As mentioned earlier, Karger introduced the influential random contraction approach for graphs leading to a randomized algorithm that runs in time $\tilde{O}(mn^2)$. Karger and Stein refined this approach to obtain a run-time of $\tilde{O}(n^2)$. Subsequently, Karger gave a tree-packing approach that runs in time $O(m \log^3 n)$. The tree-packing approach has garnered much attention recently leading to run-time improvements [12, 13] and extensions to solve more general partitioning problems [71, 106].

We now discuss the case of hypergraphs. We first focus on bounded rank hypergraphs. Fukunaga [107] generalized the tree-packing approach to r-rank hypergraphs to design an algorithm that runs in time $\tilde{O}_r(m^2 n^{2r-1})$. As mentioned earlier, Kogan and Krauthgamer [20] generalized the random contraction approach to r-rank hypergraphs to design an algorithm that runs in time $O_r(mn^2)$. Fox, Panigrahi, and Zhang [36] gave a randomized algorithm that runs in time $O(p + n^r \log^2 n)$. We now turn to arbitrary rank hypergraphs. As mentioned earlier, the approach of Nagamochi and Ibaraki was generalized in three different directions for arbitrary rank hypergraphs [31, 32, 33] with all of them leading to a deterministic run-time of $O(pn)$. Chekuri and Xu improved this deterministic run-time to $O(p+\lambda n^2)$ for unweighted hypergraphs. In a series of works [34, 35, 36], the random contraction approach was refined and extended leading to a randomized run-time of $O(mn^2 \log^3 n)$. Interestingly, randomized approaches are slower than deterministic approaches for arbitrary rank hypergraphs.

Motivated towards improving the $\log^3 n$ factor in the deterministic run-time for global min-cut in graphs, as well as obtaining a faster (randomized) algorithm, Kawarabayashi and Thorup [57, 94] initiated the study of global min-cut in simple unweighted graphs (i.e., computing connectivity). They gave a deterministic algorithm for computing connectivity that runs in time $O(m \log^{12} n)$. Henzinger, Rao, and Wang [58] improved the deterministic run-time to $O(m \log^2 n \log\log^2 n)$. Ghaffari, Nowicki, and Thorup [59] improved the randomized run-time to $O(m \log n)$ and $O(m + n \log^3 n)$. Saranurak [60] illustrated the use of expander decomposition to compute graph connectivity in time $m^{1+o(1)}$ time. All these algorithms are based on novel structural insights on graph cuts. As mentioned earlier, our work was motivated by the question of how to generalize the structural insights for graphs to constant rank hypergraphs.

2.2 PRELIMINARIES

Let $G = (V, E)$ be a hypergraph. Let $S, T \subseteq V$ be subsets of vertices. We define

$$E[S] := \{e \in E \colon e \subseteq S\}, \tag{2.3}$$

$$E(S,T) := \{e \in E \colon e \subseteq S \cup T \text{ and } e \cap S, e \cap T \neq \emptyset\}, \text{ and} \tag{2.4}$$

$$E^o(S,T) := \{e \in E \colon e \cap S, e \cap T \neq \emptyset\}. \tag{2.5}$$

We note that $E[S]$ is the set of hyperedges completely contained in S, $E(S,T)$ is the set of hyperedges contained in $S \cup T$ and intersecting both S and T, and $E^o(S,T)$ is the set of hyperedges intersecting both S and T. With this notation, if S and T are disjoint, then $E(S,T) = E[S \cup T] - E[S] - E[T]$ and moreover, if the hypergraph is a graph, then $E(S,T) = E^o(S,T)$. A cut is a partition $(S, V \setminus S)$ where both S and $V \setminus S$ are non-empty. Let $\delta(S) := E(S, V \setminus S)$. For a vertex $v \in V$, we let $\delta(v)$ represent $\delta(\{v\})$. We define the capacity of $(S, V \setminus S)$ as $|\delta(S)|$, and call a cut a min-cut if it has minimum capacity among all cuts in G. The connectivity of G is the capacity of a min-cut in G.

We recall that the *size* of a cut $(S, V \setminus S)$ is $\min\{|S|, |V \setminus S|\}$. We emphasize the distinction between the size of a cut and the capacity of a cut: size is the cardinality of the smaller side of the cut while capacity is the number of hyperedges crossing the cut.

For a vertex $v \in V$ and a subset $S \subseteq V$, we define the degree of v by $d(v) := |\delta(v)|$ and its degree inside S by $d_S(v) := |e \in \delta(v) : e \subseteq S|$. We define $\delta := \min_{v \in V} d(v)$ to be the minimum degree in G. We define $\mathrm{vol}(S) := \sum_{v \in S} d(v)$ and $\mathrm{vol}_S(T) := \sum_{v \in T} d_S(v)$. We define the *conductance* of a set $X \subseteq V$ as $\min_{\emptyset \neq S \subsetneq X} \{ \frac{|E^o(S, X \setminus S)|}{\min\{\mathrm{vol}(S), \mathrm{vol}(X \setminus S)\}} \}$. For positive integers, $i < j$, we let $[i, j]$ represent the set $\{i, i+1, \ldots, j-1, j\}$. The following proposition will be useful while counting hyperedges within nested sets.

Proposition 2.1. Let $G = (V, E)$ be an r-rank n-vertex hypergraph and let $T \subseteq S \subseteq V$. Then,

$$|E(T, S \setminus T)| \geq \left(\frac{1}{r-1}\right)(\mathrm{vol}_S(T) - r\,|E[T]|)\,.$$

Proof. For a vertex $v \in V$ and a hyperedge $e \in E$, let $Y_{v,e} := 1$ if $v \in e$ and $e \subseteq S$, and 0 otherwise. We note that

$$\mathrm{vol}_S(T) = \sum_{v \in T} d_S(v) = \sum_{v \in T}\sum_{e \in E} Y_{v,e} = \sum_{e \in E}\sum_{v \in T} Y_{v,e} = \sum_{e \in E(T, S \setminus T)}\sum_{v \in T} Y_{v,e} + \sum_{e \subseteq T}\sum_{v \in T} Y_{v,e} \tag{2.6}$$

$$\leq \sum_{e \in E(T, S \setminus T)} (r-1) + \sum_{e \subseteq T} r = (r-1)\,|E(T, S \setminus T)| + r\,|E[T]|\,. \tag{2.7}$$

Rearranging this inequality gives the statement of the claim. QED.

For the structural theorem, we need the following folklore result showing the existence of an expander decomposition in graphs (e.g., see [101]).

Theorem 2.3 (Existential graph expander decomposition)**.** Let $G = (V, E)$ be an n-vertex graph (possibly multigraph) with m edges and let $\phi \leq 1$ be a positive real value. Then, there exists a partition $\{X_1, \ldots, X_k\}$ of the vertex set V such that the following hold:

1. $\sum_{i=1}^{k} |\delta(X_i)| = O(\phi m \log n)$ and
2. For every $i \in [k]$ and every non-empty set $S \subseteq X_i$, we have that

$$|E(S, X_i \setminus S)| \geq \phi \cdot \min\{\mathrm{vol}(S), \mathrm{vol}(X_i \setminus S)\}.$$

Our algorithm for min-cut uses several other algorithms from the literature as subroutines. We state these algorithms and their guarantees here. To begin with, we will need a constructive version of Theorem 2.3 that we state below.

Theorem 2.4 (Algorithmic graph expander decomposition [102])**.** Let $G = (V, E)$ be a graph (possibly multigraph) with m edges, and let $\phi \leq 1$ be a positive real value. Then, there exists a deterministic algorithm GraphExpanderDecomp which takes G and ϕ as input and runs in time $O(m^{1+o(1)})$ to return a partition $\{X_1, \ldots, X_k\}$ of the vertex set V such that the following hold:

1. $\sum_{i=1}^{k} |\delta(X_i)| = O(\phi m^{1+o(1)})$ and

2. For every $i \in [k]$ and every nonempty set $S \subseteq X_i$, we have that $|E(S, X_i \setminus S)| \geq \phi \cdot \min\{\text{vol}(S), \text{vol}(X_i \setminus S)\}$.

Throughout this work, $o(1)$ is with respect to the number of vertices n. The following result summarizes the vertex-ordering based algorithms for computing connectivity in hypergraphs.

Theorem 2.5. [31, 32, 33] Let G be an n-vertex hypergraph of size p (possibly multihypergraph). Then, there exists a deterministic algorithm SlowMinCut that takes G as input and runs in time $O(pn)$ to compute the min-cut capacity λ of G (and also return a min-cut).

We need a fast estimator for connectivity in hypergraphs: Matula [108] designed a linear-time $(2+\epsilon)$-approximation for min-cut in graphs. Chekuri and Xu [38] generalized Matula's approach to hypergraphs.

Theorem 2.6. [38] Let G be a hypergraph of size p (possibly multihypergraph). Then, there exists an algorithm CXApproximation that takes G as input and runs in time $O(p)$ to return a positive integer k such that $\lambda < k \leq 3\lambda$, where λ is the capacity of a min-cut in G.

For most graph optimization algorithms, it is helpful to reduce the size of the input graph without losing an optimum before running slow algorithms. A k-certificate reduces the size of the hypergraph while ensuring that every cut capacity $|\delta(C)|$ does not fall below $\min\{k, |\delta(C)|\}$. The following result on k-certificates was shown by Guha, McGregor, and Tench [109]. The size of the k-certificate was subsequently improved by Chekuri and Xu via trimming hyperedges, but the Chekuri-Xu k-certificate could be a multihypergraph. Since we need a simple hypergraph as a k-certificate, we rely on the weaker result of [109].

Theorem 2.7. [38, 109] Let $G = (V, E)$ be an n-vertex hypergraph (possibly multihypergraph) of size p, and let k be a positive integer. Then, there exists an algorithm Certificate that takes G and k as input and runs in time $O(p)$ to return a subhypergraph $G' = (V, E')$ with $\sum_{e \in E'} |e| = O(rnk)$ such that for every cut $(C, V \setminus C)$ in G, we have $|\delta_{G'}(C)| \geq \min\{k, |\delta_G(C)|\}$.

Chekuri and Xu [38] noted that the slow min-cut algorithm of Theorem 2.5 can be sped up by using the k-certificate of Theorem 2.7. We illustrate their approach in Algorithm 2.1. Our algorithm can be viewed as an improvement of Algorithm 2.1—we replace the call to SlowMinCut with a call to a faster min-cut algorithm.

We briefly discuss the correctness and run-time of Algorithm 2.1. By Theorem 2.6, we have that $k > \lambda$, so Theorem 2.7 guarantees that the min-cuts of G' will be the same as the

```
CXMinCut($G$)
    $k \leftarrow$ CXApproximation($G$)
    $G' \leftarrow$ Certificate($G, k$)
    Return SlowMinCut($G'$)
```

Algorithm 2.1: Chekuri and Xu's min-cut algorithm

min-cuts of G. Thus, SlowMinCut(G') will find a min-cut for G and the algorithm will return it. We now bound the run-time: By Theorem 2.6, CXApproximation runs in $O(p)$ time to return an estimate k that is at most 3λ. By Theorem 2.7, Certificate runs in $O(p)$ time to return a subhypergraph $G' = (V, E')$ with size $p' = \sum_{e \in E'} |e| = O(rkn) = O(r\lambda n)$. Consequently, SlowMinCut runs in time $O(r\lambda n^2)$. Thus, the run-time of Algorithm 2.1 is $O(p + r\lambda n^2)$.

Corollary 2.1. [38] Let G be an n-vertex hypergraph of size p (possibly multihypergraph). Then, there exists a deterministic algorithm CXMinCut that takes G as input and runs in time $O(p + r\lambda n^2)$ to compute the min-cut capacity λ of G (and also return a min-cut).

The next result summarizes the random contraction based algorithm for computing global min-cut in low-rank hypergraphs.

Theorem 2.8. [36] Let G be an r-rank n-vertex hypergraph of size p (possibly multihypergraph). Then, there exists a randomized algorithm FPZMinCut that takes G as input and runs in time $\tilde{O}(p + n^r)$ to return a global min-cut of G with high probability.

The next result summarizes another recent algorithm for computing global min-cut in hypergraphs via the isolating cuts technique.

Theorem 2.9. [62] Let G be an r-rank n-vertex hypergraph of size p with m hyperedges (possibly multihypergraph). Then, there exists a randomized algorithm CQMinCut that takes G as input and runs in time $\tilde{O}(\sqrt{pn(m+n)^{1.5}})$ to return a global min-cut of G with high probability.

2.3 STRUCTURAL THEOREM

We prove Theorem 2.2 in this section. We call a min-cut $(C, V \setminus C)$ *moderate-sized* if its size $\min\{|C|, |V \setminus C|\}$ is in the range $(r - \log(\lambda/4r)/\log n, (\lambda/2)^{1/r})$. In Section 2.3.1, we show that a hypergraph has no moderate-sized min-cuts. In Section 2.3.2 we show that every hypergraph has an expander decomposition—i.e., a partition of the vertex set into parts which have good expansion such that the number of hyperedges intersecting multiple parts is small. In Section 2.3.3, we define Trim and Shave operations and prove properties about these operations.

We prove Theorem 2.2 in Section 2.3.4 as follows: We assume that there are no min-cuts of small size (i.e., of size at most $r - \log(\lambda/4r)/\log n$) and bound the number of hyperedges in the union of all min-cuts. For this, we find an expander decomposition and apply the Trim and Shave operations to each part of the decomposition. Since a hypergraph cannot have moderate-sized min-cuts, and there are no small-sized min-cuts by assumption, it follows that every min-cut has large size. We use this fact to show that the result of the Trim and Shave operations is a partition of V such that (1) none of the parts intersect both sides of any min-cut and (2) the number of hyperedges crossing the parts satisfies the bound in Condition 2 of the theorem.

2.3.1 No Moderate-Sized Min-Cuts

The following lemma is the main result of this section. It shows that there are no moderate-sized min-cuts.

Lemma 2.1. Let $G = (V, E)$ be an r-rank n-vertex hypergraph with connectivity λ such that $\lambda \geq r2^{r+1}$. Let $(C, V \setminus C)$ be an arbitrary min-cut. If $\min\{|C|, |V \setminus C|\} > r - \log(\lambda/4r)/\log n$, then $\min\{|C|, |V \setminus C|\} \geq (\lambda/2)^{1/r}$.

Proof. Without loss of generality, let $|C| = \min\{|C|, |V \setminus C|\}$. Let $t := |C|$ and $s := r - \log(\lambda/4r)/\log n$. We know that $s < t$. Suppose for contradiction that $t < (\lambda/2)^{1/r}$. We will show that there exists a vertex v with $|\delta(v)| < \lambda$, thus contradicting the fact that λ is the min-cut capacity. We classify the hyperedges of G which intersect C into three types as follows:

$$E_1 := \{e \in E \colon e \subseteq C\}, \tag{2.8}$$

$$E_2 := \{e \in E \colon C \subsetneq e\}, \text{ and} \tag{2.9}$$

$$E_3 := \{e \in E \colon \emptyset \neq e \cap C \neq C \text{ and } e \cap (V \setminus C) \neq \emptyset\}. \tag{2.10}$$

We note that E_1 is the set of hyperedges that are fully contained in C, E_2 is the set of hyperedges which contain C as a proper subset, and E_3 is the set of hyperedges that intersect C but are not in $E_1 \cup E_2$. We distinguish two cases.

Case 1: Suppose $t < r$. Then, the number of hyperedges that can be fully contained in C is at most 2^r, so $|E_1| \leq 2^r$. Since $(C, V \setminus C)$ is a min-cut, we have that $\lambda = |\delta(C)| = |E_2| + |E_3|$.

We note that the number of hyperedges of size i that contain all of C is at most $\binom{n-t}{i-t}$. Hence,

$$|E_2| \leq \sum_{i=t+1}^{r} \binom{n-t}{i-t} = \sum_{i=1}^{r-t} \binom{n-t}{i} \leq \sum_{i=1}^{r-t} n^i \leq 2n^{r-t}. \tag{2.11}$$

Since each hyperedge in E_3 contains at most $t-1$ of the vertices of C, a uniform random vertex of C is in such a hyperedge with probability at most $(t-1)/t$. Therefore, if we pick a uniform random vertex from C, the expected number of hyperedges from E_3 incident to it is at most $(\frac{t-1}{t})|E_3|$. Hence, there exists a vertex $v \in C$ such that

$$|\delta(v) \cap E_3| \leq \left(\frac{t-1}{t}\right)|E_3| \leq \left(\frac{t-1}{t}\right)|\delta(C)| \leq \left(\frac{r-1}{r}\right)\lambda. \tag{2.12}$$

Combining the bounds for E_1, E_2, and E_3, we have that

$$\begin{aligned}
|\delta(v)| &= |\delta(v) \cap E_1| + |E_2| + |\delta(v) \cap E_3| && (2.13)\\
&\leq |E_1| + |E_2| + |\delta(v) \cap E_3| && (2.14)\\
&\leq 2^r + 2n^{r-t} + \left(\frac{r-1}{r}\right)\lambda && (2.15)\\
&< 2^r + 2n^{r-s} + \left(\frac{r-1}{r}\right)\lambda \qquad \text{(since } t > s\text{)} && (2.16)\\
&= 2^r + \frac{\lambda}{2r} + \left(\frac{r-1}{r}\right)\lambda \qquad \text{(since } \lambda = 4rn^{r-s}\text{)} && (2.17)\\
&= \lambda + \frac{r2^{r+1} - \lambda}{2r} && (2.18)\\
&\leq \lambda. \qquad \text{(since } \lambda \geq r2^{r+1}\text{)} && (2.19)
\end{aligned}$$

Consequently, $|\delta(v)| < \lambda$, contradicting the fact that λ is the min-cut capacity.
Case 2: Suppose $t \geq r$. Then, no hyperedge can contain C as a proper subset, so $|E_2| = 0$. For each $v \in C$, the number of hyperedges e of size i such that $v \in e \subseteq C$ is at most $\binom{t-1}{i-1}$. Hence,

$$|\delta(v) \cap E_1| \leq \sum_{i=2}^{r} \binom{t-1}{i-1} = \sum_{i=1}^{r-1} \binom{t-1}{i} \leq \sum_{i=1}^{r-1} t^i \leq 2t^{r-1}. \tag{2.20}$$

Since each hyperedge in E_3 contains at most $r-1$ of the vertices of C, a random vertex of C is in such a hyperedge with probability at most $(r-1)/t$. Therefore, if we pick a random vertex from C, the expected number of hyperedges from E_3 incident to it is at most

$(\frac{r-1}{t})|E_3|$. Hence, there exists a vertex $v \in C$ such that

$$|\delta(v) \cap E_3| \leq \left(\frac{r-1}{t}\right)|E_3| \leq \left(\frac{r-1}{t}\right)\lambda. \tag{2.21}$$

Since $t < (\lambda/2)^{1/r}$ and $t \geq r$, we have that $2t^r/\lambda < t - r + 1$. Combining this with our bounds on $|\delta(v) \cap E_1|$ and $|\delta(v) \cap E_3|$, we have that

$$|\delta(v)| = |\delta(v) \cap E_1| + |\delta(v) \cap E_3| \leq 2t^{r-1} + \left(\frac{r-1}{t}\right)\lambda = \left(r - 1 + \frac{2t^r}{\lambda}\right)\frac{\lambda}{t} < \lambda. \tag{2.22}$$

Consequently, $|\delta(v)| < \lambda$, contradicting the fact that λ is the min-cut capacity. QED.

2.3.2 Hypergraph Expander Decomposition

In this section, we prove the existence of an expander decomposition and also design an algorithm to construct it efficiently in constant rank hypergraphs.

Lemma 2.2 (Existential hypergraph expander decomposition)**.** For every r-rank n-vertex hypergraph $G = (V, E)$ with $p := \sum_{e \in E} |e|$ and every positive real value $\phi \leq 1/(r-1)$, there exists a partition $\{X_1, \ldots, X_k\}$ of the vertex set V such that the following hold:

1. $\sum_{i=1}^{k} |\delta(X_i)| = O(r\phi p \log n)$, and
2. For every $i \in [k]$ and every non-empty set $S \subset X_i$, we have that

$$|E^o(S, X_i \setminus S)| \geq \phi \cdot \min\{\text{vol}(S), \text{vol}(X_i \setminus S)\}.$$

Remark 2.1. We note that for $\phi = 1$, the trivial partition of vertex set where each part is a single vertex satisfies both conditions.

Proof. We prove this by reducing to multigraphs and then using the existence of expander decomposition in multigraphs as stated in Theorem 2.3. We create a graph $G' = (V, E')$ from G as follows: For each $e \in E$, pick $v \in e$ arbitrarily, and replace e with the set of edges $\{\{u, v\} : u \in e \setminus \{v\}\}$. That is, we replace each hyperedge of G with a star. Consequently, each hyperedge e is replaced by $|e| - 1$ edges. Therefore, $|E'| \leq \sum_{e \in E} |e| = p$. Furthermore, for every $X \subseteq V$, we have that $\text{vol}_G(X) \leq \text{vol}_{G'}(X)$ and $|\delta_G(X)| \leq |\delta_{G'}(X)|$. Also, for every $S \subseteq X$, we have that $|E^o_G(S, X \setminus S)| \geq \frac{1}{r-1}|E_{G'}(S, X \setminus S)|$.

Now, let $\{X_1, \ldots, X_k\}$ be the partition obtained by using Theorem 2.3 on input graph G' with parameter $\phi' = (r-1)\phi$. Then,

$$\sum_{i=1}^{k} |\delta_G(X_i)| \leq \sum_{i=1}^{k} |\delta_{G'}(X_i)| = O(\phi'|E'|\log n) = O(r\phi p \log n). \tag{2.23}$$

Moreover, for every $i \in [k]$ and every non-empty set $S \subset X_i$, we have that

$$\begin{aligned} |E_G^o(S, X_i \setminus S)| &\geq \left(\frac{1}{r-1}\right) |E_{G'}(S, X_i \setminus S)| & (2.24)\\ &\geq \left(\frac{\phi'}{r-1}\right) \min\{\mathrm{vol}_{G'}(S), \mathrm{vol}_{G'}(X_i \setminus S)\} & (2.25)\\ &\geq \phi \min\{\mathrm{vol}_G(S), \mathrm{vol}_G(X_i \setminus S)\}. & (2.26)\end{aligned}$$

QED.

Next, we show that it is possible to efficiently find an expander decomposition with almost the same guarantees as in the existential theorem.

Lemma 2.3 (Algorithmic hypergraph expander decomposition)**.** There exists a deterministic algorithm that takes as input an r-rank n-vertex hypergraph $G = (V, E)$ with $p := \sum_{e \in E} |e|$ and a positive real value $\phi \leq 1/(r-1)$ and runs in time $O(p^{1+o(1)})$ to return a partition $\{X_1, \ldots, X_k\}$ of the vertex set V such that the following hold:

1. $\sum_{i=1}^{k} |\delta(X_i)| = O(r\phi p^{1+o(1)})$, and
2. For every $i \in [k]$, and for every non-empty set $S \subset X_i$, we have that $|E^o(S, X_i \setminus S)| \geq \phi \cdot \min\{\mathrm{vol}(S), \mathrm{vol}(X_i \setminus S)\}$.

Proof. The proof is similar to that of Lemma 2.2—via a reduction to graphs and using the result of [102] mentioned in Theorem 2.4 (which is a constructive version of Theorem 2.3). We note that Theorem 2.4 gives a worse guarantee on $\sum_{i=1}^{k} |\delta_G(X_i)|$ than that in Theorem 2.3. Hence, we revisit the calculations for Condition 1 below. The notations that we use below are borrowed from the proof of Lemma 2.2.

Let $\{X_1, \ldots, X_k\}$ be the partition returned by algorithm GraphExpanderDecomp (from Theorem 2.4) on input graph G' with parameter $\phi' = (r-1)\phi$. Then,

$$\sum_{i=1}^{k} |\delta_G(X_i)| \leq \sum_{i=1}^{k} |\delta_{G'}(X_i)| \leq = O\left(\phi'|E'|^{1+o(1)}\right) = O\left(r\phi p^{1+o(1)}\right). \tag{2.27}$$

The graph G' in the proof of Lemma 2.2 (also used in this proof) can be constructed in time $O(p)$. The algorithm GraphExpanderDecomp from Theorem 2.4 on input graph G' runs in time $O(p^{1+o(1)})$. Hence, the total running time of this algorithm is $O(p^{1+o(1)})$. QED.

2.3.3 Trim and Shave Operations

In this section, we define the Trim and Shave operations and prove certain useful properties about them.

Definition 2.1. Let $G = (V, E)$ be an r-rank hypergraph. For $X \subseteq V$, let

1. $\textsc{Trim}(X)$ be the set obtained by repeatedly removing from X a vertex v with $d_X(v) < d(v)/2r$ until no such vertices remain,
2. $\textsc{Shave}(X) := \{v \in X : d_X(v) > (1 - 1/r^2)d(v)\}$, and
3. $\textsc{Shave}_k(X) := \textsc{Shave}(\textsc{Shave} \cdots (\textsc{Shave}(X)))$ be the result of applying k consecutive Shave operations to X.

We emphasize that Trim is an adaptive operation while Shave is a non-adaptive operation and $\textsc{Shave}_k(X)$ is a sequence of Shave operations. We now prove certain useful properties about these operations. In the rest of this section, let $G = (V, E)$ be an r-rank n-vertex hypergraph with minimum degree δ and min-cut capacity λ. The following claim shows that the Trim operation on a set X that has small intersection with a min-cut further reduces the intersection.

Claim 2.1. Let $(C, V \setminus C)$ be a min-cut. Let X be a subset of V and $X' := \textsc{Trim}(X)$. If $\min\{|X \cap C|, |X \cap (V \setminus C)|\} \le (\delta/6r^2)^{1/(r-1)}$, then

$$\min\{|X' \cap C|, |X' \cap (V \setminus C)|\} \le 3r^2.$$

Proof. Without loss of generality, we assume that $|X \cap C| = \min\{|X \cap C|, |X \cap (V \setminus C)|\}$. We have that

$$\begin{aligned}
\delta &\ge \lambda && (2.28)\\
&\ge |E(X' \cap C, X' \cap (V \setminus C))| && (2.29)\\
&\ge \left(\frac{1}{r-1}\right)\left(\mathrm{vol}_{G[X']}(X' \cap C) - r|E(X' \cap C, X' \cap C)|\right) \qquad \text{(By Proposition 2.1)} && (2.30)\\
&\ge \left(\frac{1}{r-1}\right)\left(\frac{1}{2r}\delta|X' \cap C| - r|X' \cap C|^r\right). && (2.31)
\end{aligned}$$

The last inequality above is by definition of Trim. Rearranging the above, we obtain that

$$r|X' \cap C|^r \geq \left(\frac{1}{2r}|X' \cap C| - (r-1)\right)\delta. \tag{2.32}$$

Dividing by $|X' \cap C|$, we have that

$$r|X' \cap C|^{r-1} \geq \left(\frac{1}{2r} - \frac{(r-1)}{|X' \cap C|}\right)\delta. \tag{2.33}$$

We note that $|X' \cap C| \leq |X \cap C| = \min\{|X \cap C|, |X \cap (V \setminus C)|\} \leq (\delta/6r^2)^{1/(r-1)}$. Therefore,

$$\frac{\delta}{6r} \geq \left(\frac{1}{2r} - \frac{r-1}{|X' \cap C|}\right)\delta. \tag{2.34}$$

Dividing by δ and rearranging, we obtain that

$$|X' \cap C| \leq 3r(r-1) \leq 3r^2. \tag{2.35}$$

QED.

The following claim shows that the Shave operation on a set X which has small intersection with a large-sized min-cut further reduces the intersection.

Claim 2.2. Suppose $\lambda \geq r(4r^2)^r$. Let $(C, V \setminus C)$ be a min-cut with $\min\{|C|, |V \setminus C|\} \geq 4r^2$. Let X' be a subset of V and $X'' := \text{Shave}(X')$. If $0 < \min\{|X' \cap C|, |X' \cap (V \setminus C)|\} \leq 3r^2$, then

$$\min\{|X'' \cap C|, |X'' \cap (V \setminus C)|\} \leq \min\{|X' \cap C|, |X' \cap (V \setminus C)|\} - 1.$$

Proof. Without loss of generality, we assume that $|X' \cap C| = \min\{|X' \cap C|, |X' \cap (V \setminus C)|\}$. Since $X'' \subseteq X'$, we have that $|X'' \cap C| \leq |X' \cap C|$. Thus, we only need to show that this inequality is strict. Suppose for contradiction that $|X'' \cap C| = |X' \cap C|$. We note that $0 < |X'' \cap C| \leq 3r^2$.

Let $Z := X' \cap C = X'' \cap C$, and let $C' := C - X'$. Since $|C| \geq \min\{|C|, |V \setminus C|\} \geq 4r^2$ and $|Z| \leq 3r^2$, we know that C' is nonempty.

We note that $Z \subseteq X''$. By definition of Shave, we have that

$$\text{vol}_{X'}(Z) = \sum_{v \in Z} d_{X'}(v) > \sum_{v \in Z}\left(1 - \frac{1}{r^2}\right)d(v) = \left(1 - \frac{1}{r^2}\right)\text{vol}(Z). \tag{2.36}$$

We note that $|E(Z, V \setminus C)| \geq |E(Z, X' \setminus C)| = |E(Z, X' \setminus Z)|$, so by Proposition 2.1, we have that

$$|E(Z, V \setminus C)| \geq |E(Z, X' \setminus Z)| \tag{2.37}$$

$$\geq \left(\frac{1}{r-1}\right)(\mathrm{vol}_{X'}(Z) - r|E[Z]|) \tag{2.38}$$

$$> \left(\frac{1}{r-1}\right)\left(\left(1 - \frac{1}{r^2}\right)\mathrm{vol}(Z) - r|Z|^r\right). \tag{2.39}$$

We also know from the definition of Shave that

$$|E(Z, C \setminus Z)| \leq \sum_{v \in Z} |E(\{v\}, C \setminus Z)| \leq \sum_{v \in Z} \frac{1}{r^2} d(v) = \frac{\mathrm{vol}(Z)}{r^2}. \tag{2.40}$$

Therefore, using our assumption that $\lambda \geq r(4r^2)^r$, we have that

$$|E(Z, (V \setminus C))| > \left(\frac{1}{r-1}\right)\left(\left(1 - \frac{1}{r^2}\right)\mathrm{vol}(Z) - r|Z|^r\right) \tag{2.41}$$

$$= \left(\frac{r^2 - 1}{r^2(r-1)}\right)\mathrm{vol}(Z) - \left(\frac{r}{r-1}\right)|Z|^r \tag{2.42}$$

$$= \frac{\mathrm{vol}(Z)}{r^2} + \frac{\mathrm{vol}(Z)}{r} - \left(\frac{r}{r-1}\right)|Z|^r \tag{2.43}$$

$$= \frac{\mathrm{vol}(Z)}{r^2} + \frac{\sum_{v \in Z} d(v)}{r} - r|Z|^r \tag{2.44}$$

$$\geq \frac{\mathrm{vol}(Z)}{r^2} + \frac{|Z|\lambda}{r} - r|Z|^r \tag{2.45}$$

$$\geq \frac{\mathrm{vol}(Z)}{r^2} + (4r^2)^r|Z| - r|Z|^r \tag{2.46}$$

$$\geq \frac{\mathrm{vol}(Z)}{r^2} + (4r^2)|Z|^r - r|Z|^r \tag{2.47}$$

$$\geq \frac{\mathrm{vol}(Z)}{r^2} \tag{2.48}$$

$$\geq |E(Z, C \setminus Z)|. \tag{2.49}$$

We note that $E(Z, (V \setminus C))$ is the set of hyperedges which are cut by C but not C', while $E(Z, C \setminus Z)$ is the set of hyperedges which are cut by C' but not C. Since we have shown that $|E(Z, V \setminus C)| > |E(Z, C \setminus Z)|$, we conclude that $|\delta(C)| > |\delta(C')|$. Since $(C, V \setminus C)$ is a min-cut and $\emptyset \neq C' \subseteq C \subsetneq V$, this is a contradiction. QED.

The following claim shows that the Trim operation increases the cut capacity by at most a constant factor.

Claim 2.3. Let X be a subset of V and $X' := \textsc{Trim}(X)$. Then,

1. $|E[X] - E[X']| \leq |\delta(X)|$, and
2. $|\delta(X')| \leq 2|\delta(X)|$.

Proof. We will prove the first part of the claim. The second part follows from the first part.

For a set $S \subseteq V$, we define a potential function $f(S) := \sum_{e \in \delta(S)} |e \cap S|$. Since every hyperedge in $\delta(S)$ must include at least 1 and at most $r-1$ vertices of S, we have that $|\delta(S)| \leq f(S) \leq (r-1)|\delta(S)|$. We note that the removal of a vertex v from X during the Trim process decreases $f(X)$ by at least $(1 - \frac{1}{2r})d(v) - \frac{(r-1)}{2r}d(v) = \frac{1}{2}d(v)$. This is because, from the definition of Trim, at least a $1 - \frac{1}{2r}$ fraction of v's hyperedges were part of $\delta(X)$ before v was removed, and each of these hyperedges losing a vertex causes $f(X)$ to decrease by one, while v's removal can add at most $\frac{1}{2r}d(v)$ new hyperedges to $\delta(X)$, and each of these hyperedges can contribute at most $r-1$ to $f(X)$. We also note that when v is removed, the cut capacity $|\delta(X)|$ increases by at most $|E(\{v\}, X \setminus \{v\})| \leq \frac{1}{2r}d(v)$. Therefore, the removal of a vertex v from X during the Trim process increases the cut capacity $|\delta(X)|$ by at most $\frac{1}{2r}d(v)$ and decreases the potential function $f(X)$ by at least $\frac{1}{2}d(v)$. Thus, for every additional hyperedge added to $\delta(X)$, the potential function must decrease by at least $\frac{d(v)/2}{d(v)/2r} = r$. Before the Trim operation, we have $f(X) \leq (r-1)|\delta(X)|$, so the number of hyperedges that can be removed from X at the end of the Trim operation is at most $\frac{1}{r}(r-1)|\delta(X)| \leq |\delta(X)|$. QED.

The following claim shows that the Shave operation increases the cut capacity by at most a factor of r^3.

Claim 2.4. Let X' be a subset of V and $X'' := \textsc{Shave}(X')$. Then

1. $|E[X'] - E[X'']| \leq r^2(r-1)|\delta(X')|$, and
2. $|\delta(X'')| \leq r^3|\delta(X')|$.

Proof. We will prove the first part of the claim. The second part follows from the first part.

We note that the removal of a vertex v from X' during the Shave process decreases the number of hyperedges fully contained in X by at most $d(v)$. We also know that if v is removed, we must have $|\delta(v) \cap \delta(X')| \geq d(v)/r^2$, and therefore $d(v) \leq r^2|\delta(v) \cap \delta(X')|$. Thus, the total number of hyperedges removed from X' in a single Shave operation is at most $r^2 \sum_{v \in X'} |\delta(v) \cap \delta(X')| \leq r^2(r-1)|\delta(X')|$. QED.

2.3.4 Proof of Theorem 2.2

In this section, we restate and prove Theorem 2.2.

Theorem 2.2. [Structure] Let $G = (V, E)$ be an r-rank n-vertex simple hypergraph with m hyperedges and connectivity λ. Suppose $\lambda \geq r(4r^2)^r$. Then, at least one of the following holds:

1. There exists a min-cut $(C, V \setminus C)$ such that
$$\min\{|C|, |V \setminus C|\} \leq r - \frac{\log\left(\frac{\lambda}{4r}\right)}{\log n},$$
2. The number of hyperedges in the union of all min-cuts is
$$O\left(r^{9r^2+2}\left(\frac{6r^2}{\lambda}\right)^{\frac{1}{r-1}} m \log n\right) = \tilde{O}_r\left(\frac{m}{\lambda^{\frac{1}{r-1}}}\right).$$

Proof. Suppose the first conclusion does not hold. Then, by Lemma 2.1, the smaller side of every min-cut has size at least $(\lambda/2)^{1/r} \geq 4r^2$. Let $(C, V \setminus C)$ be an arbitrary min-cut. We use Lemma 2.2 with $\phi = (6r^2/\lambda)^{1/(r-1)}$ to get an expander decomposition $\mathcal{X} = \{X_1, \ldots, X_k\}$. We note that $\phi \leq 1/(r-1)$ holds by the assumption that $\lambda \geq r(4r^2)^r$. For $i \in \{1, \ldots, k\}$, let $X_i' := \textsc{Trim}(X_i)$ and $X_i'' := \textsc{Shave}_{3r^2}(X_i')$.

Let $i \in [k]$. By the definition of the expander decomposition and our choice of $\phi = (6r^2/\lambda)^{1/(r-1)}$, we have that

$$\lambda \geq |E^o(X_i \cap C, X_i \cap (V \setminus C))| \tag{2.50}$$
$$\geq \left(\frac{6r^2}{\lambda}\right)^{\frac{1}{r-1}} \min\{\text{vol}(X_i \cap C), \text{vol}(X_i \setminus C)\} \tag{2.51}$$
$$\geq \left(\frac{6r^2}{\lambda}\right)^{\frac{1}{r-1}} \delta \min\{|X_i \cap C|, |X_i \setminus C|\}. \tag{2.52}$$

Thus, $\min\{|X_i \cap C|, |X_i \setminus C|\} \leq (\lambda/\delta)(\lambda/6r^2)^{1/(r-1)} \leq (\lambda/6r^2)^{1/(r-1)} \leq (\delta/6r^2)^{1/(r-1)}$.

Therefore, by Claim 2.1, we have that $\min\{|X_i' \cap C|, |X_i' \cap (V \setminus C)|\} \leq 3r^2$. We recall that $\lambda \geq r(4r^2)^r$ and every min-cut has size at least $4r^2$. By $3r^2$ repeated applications of Claim 2.2, we have that $\min\{|X_i'' \cap C|, |X_i'' \cap (V \setminus C)|\} = 0$.

Let $\mathcal{X}'' := \{X_1'', \ldots, X_k''\}$. Since $\min\{|X_i'' \cap C|, |X_i'' \cap (V \setminus C)|\} = 0$ for every min-cut $(C, V \setminus C)$ and every $X_i'' \in \mathcal{X}''$, it follows that no hyperedge crossing a min-cut is fully contained within a single part of $\mathcal{X}''$. Thus, it suffices to show that $|E - \bigcup_{i=1}^k E[X_i'']|$ is small—i.e., the number of hyperedges not contained in any of the parts of $\mathcal{X}''$ is $\tilde{O}_r(m/\lambda^{\frac{1}{r-1}})$.

By Claim 2.3, we have that $|E[X_i] - E[X_i']| \le 2|\delta(X_i)|$ and $|\delta(X_i')| \le 2|\delta(X_i)|$ for each $i \in [k]$. By the second part of Claim 2.4, we have that $|\delta(\textsc{Shave}_{j+1}(X_i'))| \le r^3|\delta(\textsc{Shave}_j(X_i'))|$ for every non-negative integer j. Therefore, by repeated application of the second part of Claim 2.4, for every $j \in [3r^2]$, we have that $|\delta(\textsc{Shave}_j(X_i'))| \le 2r^{3j}|\delta(X_i)|$. By the first part of Claim 2.4, for every $j \in [3r^2]$, we have that $|E[\textsc{Shave}_{j-1}(X_i')] - E[\textsc{Shave}_j(X_i')]| \le r^3|\delta(\textsc{Shave}_{j-1}(X_i'))| \le 2r^{3j}|\delta(X_i)|$. Therefore,

$$\left|E - \bigcup_{i=1}^k E[X_i'']\right| \tag{2.53}$$

$$\le \left|E - \bigcup_{i=1}^k E[X_i]\right| + \sum_{i=1}^k |E[X_i] - E[X_i'']| \tag{2.54}$$

$$= \left|E - \bigcup_{i=1}^k E[X_i]\right| + \sum_{i=1}^k \left(|E[X_i] - E[X_i']| + \sum_{j=1}^{3r^2} |E[\textsc{Shave}_{j-1}(X_i')] - E[\textsc{Shave}_j(X_i')]|\right) \tag{2.55}$$

$$\le \left|E - \bigcup_{i=1}^k E[X_i]\right| + \sum_{i=1}^k \left(2|\delta(X_i)| + \sum_{j=1}^{3r^2} 2r^{3j}|\delta(X_i)|\right) \tag{2.56}$$

$$= \left|E - \bigcup_{i=1}^k E[X_i]\right| + \sum_{i=1}^k |\delta(X_i)| \left(2 + \sum_{j=1}^{3r^2} 2r^{3j}\right) \tag{2.57}$$

$$\le \left|E - \bigcup_{i=1}^k E[X_i]\right| + \sum_{i=1}^k 3r^{9r^2}|\delta(X_i)| \tag{2.58}$$

$$\le \left|E - \bigcup_{i=1}^k E[X_i]\right| + 3r^{9r^2}\sum_{i=1}^k |E(X_i, V \setminus X_i)| \tag{2.59}$$

$$\le 4r^{9r^2}\sum_{i=1}^k |E(X_i, V \setminus X_i)|\,. \tag{2.60}$$

By Lemma 2.2, since $\mathcal{X}$ is an expander decomposition for $\phi = (6r^2/\lambda)^{1/(r-1)}$ and since $p = \sum_{e \in E}|e| \le mr$, we have that

$$\sum_{i=1}^k |E(X_i, V \setminus X_i)| = O(r\phi p \log n) = O(r)\left(\frac{6r^2}{\lambda}\right)^{\frac{1}{r-1}} p \log n = O(r^2)\left(\frac{6r^2}{\lambda}\right)^{\frac{1}{r-1}} m \log n. \tag{2.61}$$

Thus, $|E - \bigcup_{i=1}^k E[X_i'']| = O(r^{9r^2+2}(6r^2/\lambda)^{1/(r-1)} m \log n)$, thus proving the second conclusion.

QED.

2.4 ALGORITHM

In this section, we prove Theorem 2.1. In Section 2.4.1, we give an algorithm to find a min-cut in hypergraphs where some min-cut has small size. This algorithm uses either the algorithm we give in Section 2.4.2, or the algorithm that we give in Section 2.4.3, depending on the min-cut value of the input hypergraph. In Section 2.4.4, we give an algorithm which uses the expander decomposition and the TRIM and SHAVE operations defined in Section 2.3.3 to find a min-cut if the size of every min-cut is large. In Section 2.4.5, we apply the sparsification algorithm of Chekuri and Xu, followed by a combination of our two algorithms in order to prove Theorem 2.1.

2.4.1 Small-Sized Hypergraph Min-Cut

We recall that the size of a cut is the number of vertices on the smaller side of the cut. In this section, we give a randomized algorithm to find connectivity in hypergraphs in which some min-cut has size at most s. The following is the main result of this section.

Theorem 2.10. Let s be a positive integer and let $G = (V, E)$ be an r-rank hypergraph of size p that has a min-cut $(C, V \setminus C)$ with $|C| \leq s$. Then, there exists a randomized algorithm SMALLSIZEMINCUT which takes G and s as input and runs in time $\tilde{O}_r(s^{6r}p)$ to return a min-cut of G with high probability.

Remark 2.2. We note that the above result gives a randomized algorithm. In contrast, we give a simple but slightly slower deterministic algorithm for this problem in Section 2.5.

We prove Theorem 2.10 using one of two possible algorithms based on the connectivity λ of the input hypergraph: By Theorem 2.6, we can find a k such that $\lambda \leq k \leq 3\lambda$ in $O(p)$ time. If $k \leq 2700s^r \log n$, then we use the algorithm from Section 2.4.2 to find a min-cut in time $\tilde{O}_r(s^{3r}p + s^{6r})$ and if $k > 2700s^r \log n$ (i.e., $\lambda > 900s^r \log n$), then we use the algorithm from Section 2.4.3 to find a min-cut in time $\tilde{O}_r(s^{3r}p)$. These two algorithms exploit two slightly different ways of representing the hypergraph as a graph.

2.4.2 Small-Sized Min-Cut for Small λ

This section is devoted to proving the following theorem, which allows us to find a min-cut in a hypergraph with a small-sized min-cut. The algorithm is fast if the min-cut capacity is sufficiently small.

Theorem 2.11. Let s be a positive integer and let $G = (V, E)$ be an r-rank hypergraph of size p that has a min-cut $(C, V \setminus C)$ with $|C| \leq s$. Then, there exists a randomized algorithm which takes G and s as input and runs in time $\tilde{O}_r(p\lambda^2 rs^r + \lambda^4 r^2 s^{2r})$ to return a min-cut of G with high probability.

In the rest of this section, let s be a positive integer and let $G = (V, E)$ be an r-rank n-vertex hypergraph of with m hyperedges and size p that has a min-cut $(C, V \setminus C)$ with $|C| \leq s$. Let λ be the min-cut capacity of G and let k be the approximation of λ obtained by running the algorithm from Theorem 2.6. If $p \leq 512k^2 rs^r$, then $n \leq p \leq 512k^2 rs^r$. In this case, we run the algorithm SlowMinCut from Theorem 2.5 to compute the min-cut in G in time $O(np) = \tilde{O}(\lambda^4 r^2 s^{2r})$. Henceforth, we assume that $p > 512k^2 rs^r$.

We construct an arc-weighted directed graph $D_G = (V_D, E_D)$ as follows: The vertex and arc sets of D_G are given by $V_D := U_V \cup U_{E_{in}} \cup U_{E_{out}}$ and $E_D := E_1 \cup E_2$ respectively, where $U_V, U_{E_{in}}, U_{E_{out}}, E_1$, and E_2 are defined next. The vertex set U_V has a vertex u_v for each $v \in V$. The vertex set $U_{E_{in}}$ has a vertex e_{in} for every hyperedge $e \in E$, and the vertex set $U_{E_{out}}$ has a vertex e_{out} for every hyperedge $e \in E$. The arc set E_1 has an arc (e_{in}, e_{out}) for every $e \in E$. The arc set E_2 has arcs (v, e_{in}) and (e_{out}, v) for every $e \in E$ and $v \in e$. We assign a weight of 1 to each arc in E_1 and a weight of ∞ to each arc in E_2. We note that $|V_D| = n + 2m$, and $|E_D| = m + 2p$. Furthermore, D_G can be constructed from G in $\tilde{O}(p)$ time.

Definition 2.2. A directed cut in D_G is an ordered partition of the vertices into two parts $(C, V_D \setminus C)$ such that $\emptyset \neq C \cap U_V \neq U_V$, and its weight is the total weight of arcs outgoing from C.

The following proposition relates min-cut in G to min-weight directed cut in D_G.

Proposition 2.2. An algorithm that runs in time $\tilde{O}(f(p, s, r, \lambda))$ to find a min-weight directed cut in D_G, for some function f, can be used to find a min-cut in G in time $\tilde{O}(p + f(p, s, r, \lambda))$.

Proof. We first note that every cut $(C, V \setminus C)$ in G has a corresponding cut in D_G, namely $(C', V_D \setminus C')$ where $C' := \{u_v \colon v \in C\} \cup \{e_{in} \colon e \cap C \neq \emptyset\} \cup \{e_{out} \colon e \subseteq C\}$. The arcs cut by $(C', V_D \setminus C')$ will be arcs of the form (e_{in}, e_{out}) where $e \cap C \neq \emptyset$, but e is not contained in C, and the number of such arcs is exactly the number of hyperedges in $\delta(C)$. Therefore, the min-weight of a directed cut in D_G is at most the min-cut capacity in G.

Next, let $(C', V_D \setminus C')$ be a min-weight directed cut in D_G. Let $C := \{v \colon u_v \in C'\}$. We will show that $C' \cap U_{E_{in}} = \{e_{in} \colon e \cap C \neq \emptyset\}$ and $C' \cap U_{E_{out}} = \{e_{out} \colon e \subseteq C\}$. We note that a min-cut in D_G does not cut any infinite cost hyperedges. Therefore, C' contains every vertex e_{in} such that $e \cap C \neq \emptyset$. Similarly, C' can contain a vertex e_{out} only if $e \subseteq C$. Thus, $(C', V_D \setminus C')$ cuts all arcs (e_{in}, e_{out}) where $e \in \delta(C)$. Therefore, the min-weight of a directed cut in D_G is at least the min-cut capacity in G.

The above arguments imply that the min-weight of a directed cut in D_G is equal to the min-cut capacity in G and moreover, given a min-weight directed cut in D_G, we can recover a min-cut in G immediately. The run-time guarantees follow.

QED.

By Proposition 2.2, in order to prove Theorem 2.11, it suffices to design an algorithm that runs in time $\tilde{O}_r(p\lambda^2 rs^r + \lambda^4 r^2 s^{2r})$ to find a min-weight directed cut $(C, V_D \setminus C)$ in the directed graph D_G corresponding to G, under the assumptions that $|C \cap U_V| \leq s$ and $p \geq 512k^2rs^r$. In Algorithm 2.2 we give a randomized algorithm to find a min-weight directed cut under these assumptions. This algorithm is inspired by an algorithm of Forster, Nanongkai, Yang, Saranurak and Yingchareonthawornchai [96]. We show in Theorem 2.12 that if $x \in C$ for some min-weight directed cut $(C, V_D \setminus C)$ in D_G with $|C \cap U_V| \leq s$, then running Algorithm 2.2 on (D_G, x, k, s) will return a min-weight directed cut with constant probability. Thus, running Algorithm 2.2 on (D_G, x, k, s) for each $x \in U_V$ and returning the best cut found will indeed return a min-weight directed cut in D_G with constant probability; repeating this process $\log n$ times and taking the best cut found will return a min-weight directed cut with high probability. We also show in Theorem 2.12 that Algorithm 2.2 can be implemented to run in time $O(\lambda^3 rs^r)$. Thus, running SmallSizeSmallMinCut $O(\log n)$ times for each $x \in U_V$ and returning the best cut found takes time $\tilde{O}(nk^3rs^r) = \tilde{O}(pk^2rs^r)$. Adding in the time for the base case where $p \leq 512k^2rs^r$, we get a total running time of $\tilde{O}(p\lambda^2rs^r + \lambda^4r^2s^{2r})$, thus proving Theorem 2.12.

Theorem 2.12. Let $(C, V_D \setminus C)$ be a min-weight directed cut in D_G with $|C \cap U_V| \leq s$ and $p \geq 512k^2rs^r$. For $x \in C$, Algorithm 2.2 returns a min-weight directed cut in D_G with probability at least $\frac{3}{4}$. Moreover, the algorithm can be implemented to run in time $O(\lambda^3rs^r)$.

Proof. We begin by showing the following claim.

Claim 2.5. If Algorithm 2.2 returns $V(T)$ in iteration j of its for-loop, then $(V(T), V_D \setminus V(T))$ is a directed cut in D_G with weight at most $j - 1$.

Proof. We note that whenever we flip the arcs in an $x - y$ path, the weight of every directed cut $(C, V_D \setminus C)$ with $x \in C$ either decreases by 1 (if $y \in V_D \setminus C$) or stays the same. We also

```
SmallSizeSmallMinCut($D_G, x, k, s$)
  Set all arcs as unmarked
  For $i$ from 1 to $k+1$
      $y \leftarrow NIL$
      Run a new iteration of BFS from $x$ to get a BFS tree $T$ as follows:
      While the BFS algorithm still has an arc $(u,v)$ to explore
          If $(u,v)$ is not marked
              Mark $(u,v)$
              If at least $512k^2rs^r$ arcs are marked
                  Return $\perp$
              With probability $\frac{1}{8r(4s^r+k)}$, stop BFS and set $y \leftarrow v$
      If $y = NIL$
          Return $V(T)$
      Flip the orientation of all arcs on the $x-y$ path in $T$
  Return $\perp$
```

Algorithm 2.2: Algorithm to find a min-weight directed cut

note that if the algorithm eventually returns $V(T)$, then the last iteration must have explored all arcs reachable from x without exploring $512k^2rs^r$ arcs. Since $|E_D| \geq p > 512k^2rs^r$, this means that not all arcs of D_G are reachable from x, and therefore $(V(T), V_D \setminus V(T))$ is a directed cut in D_G. The weight of the cut $(V(T), V_D \setminus V(T))$ has decreased by at most 1 in each previous iteration of BFS run, since this is the only time when the algorithm flips the arcs of a path. Therefore, if $V(T)$ is returned in iteration j of the algorithm's for-loop, the corresponding cut in D_G has weight at most $j-1$. QED.

By Claim 2.5, it suffices to show that with constant probability, Algorithm 2.2 returns $V(T)$ after $\lambda+1$ iterations of BFS. By Claim 2.5, the algorithm cannot return $V(T)$ within λ iterations of BFS (since this would imply that there exists a directed cut with weight less than λ). Therefore, since $\lambda \leq k$, the algorithm will always run at least $\lambda+1$ iterations of BFS if it does not terminate because of marking $512k^2rs^r$ arcs. If $x \in C$, then whenever the algorithm flips the orientations of all arcs of an $x-y$ path where $y \in V_D \setminus C$, the weight of the cut $(C, V_D \setminus C)$ in the resulting graph decreases by 1. Thus, if $x \in C$ and the algorithm fails to return a min-cut after $\lambda+1$ iterations, either the first $\lambda+1$ iterations marked more than $512k^2rs^r$ arcs, or one of the choices for vertex y chosen within the first λ iterations is in C.

Claim 2.6. The probability that Algorithm 2.2 returns $\perp$ due to marking at least $512k^2rs^r$ arcs within the first $\lambda+1$ iterations of BFS is at most 1/8.

Proof. The expected number of arcs marked by a single iteration of BFS is $8r(4s^r+k)$, so the expected number of arcs marked by $\lambda + 1$ iterations of BFS is

$$8(\lambda + 1)r(4s^r + k) \leq 8(k + 1)r(4s^r + k) \leq 64rk^2s^r. \tag{2.62}$$

Therefore, by Markov's inequality, the probability that at least $512k^2rs^r$ arcs are marked in the first $\lambda + 1$ iterations is at most $1/8$. QED.

Claim 2.7. The probability that Algorithm 2.2 picks a vertex from C as y in some iteration of its execution is at most $1/8$.

Proof. If the algorithm never terminates a BFS iteration on an arc with an endpoint in C, then it will always choose the vertex y from $V_D \setminus C$. We recall that $|C \cap U_V| \leq s$. Therefore, $|C \cap U_{e_{out}}| \leq 2s^r$, since the number of hyperedges of size at most r over a subset of vertices of size s is at most

$$\binom{s}{r} + \binom{s}{r-1} + \binom{s}{r-2} + \cdots + \binom{s}{2} \leq s^r + \cdots + s^2 \leq 2s^r. \tag{2.63}$$

Since C is a min-weight directed cut, we also have that $|C \cap U_{e_{in}}| \leq |C \cap U_{e_{out}}| + \lambda \leq |C \cap U_{e_{out}}| + k$. Also, since C is a min-weight directed cut, we know that for an arbitrary vertex v in $C \cap U_V$, all of v's neighbors are also in C (since the arcs between vertices in U_V and $V_D \setminus U_V$ have infinite weight). Thus, every arc which is incident to a vertex in C is incident to a vertex in $C \cap (U_{e_{in}} \cup U_{e_{out}})$. Since each vertex in $U_{e_{in}} \cup U_{e_{out}}$ has degree at most r, this means that the total number of arcs which touch vertices in C is at most $r(|C \cap U_{e_{in}}| + |C \cap U_{e_{out}}|) \leq r(4s^r + k)$. Each of these arcs is marked at most once during the execution of the algorithm, and has a $1/(8r(4s^r + k))$ chance of causing BFS to terminate when it is marked. Thus, by union bound, the probability that one of these arcs causes the BFS to terminate (and potentially not flip an arc from the min-cut) is at most $1/8$. QED.

Combining Claim 2.6 with Claim 2.7, and by union bound, the total probability that the algorithm returns an incorrect answer is at most $1/4$.

Finally, we analyze the runtime of Algorithm 2.2. The runtime is dominated by the breadth first searches. There are at most $k + 1$ searches, and each breadth first search visits at most $512k^2rs^r$ arcs, so the runtime of Algorithm 2.2 is $O(\lambda^3rs^r)$. QED.

2.4.3 Small-Sized Min-Cut for Large λ

In this section, we prove the following theorem which allows us to find min-cut in hypergraphs where some min-cut has small size. The algorithm is fast if the min-cut capacity is sufficiently large.

Theorem 2.13. Let s be a positive integer, and let $G = (V, E)$ be an n-vertex r-rank hypergraph of size p with $n \geq 2r$ that has a min-cut $(C, V \setminus C)$ with $|C| \leq s$ and min-cut capacity $\lambda \geq 900s^r \log n$. Then, there exists a randomized algorithm which takes G and s as input and runs in time $\tilde{O}_r(s^{3r}p)$ to return a min-cut of G with high probability.

In the rest of this section, let s be a positive integer and let $G = (V, E)$ be an r-rank n-vertex hypergraph of with m hyperedges and size p that has a min-cut $(C, V \setminus C)$ with $|C| \leq s$ and min-cut capacity $\lambda \geq 900s^r \log n$. We construct a vertex-weighted bipartite graph $B = (V_B, E_B)$ as follows: The vertex set of B is given by $V_B := U_V \cup U_E$, and the edge set of B is E_B, where U_V, U_E, E_B are defined next. The vertex set U_V has a vertex u_v for every vertex $v \in V$. The vertex set U_E has a vertex u_e for every hyperedge $e \in E$. The edge set E_B has an edge $\{u_v, u_e\}$ for every $v \in V$, $e \in E$ such that the hyperedge e contains the vertex v in G. We assign a weight of ∞ to each vertex in U_V and a weight of 1 to each vertex in U_E. We note that $|V_B| = m + n$ and $|E_B| = p$. Furthermore, the graph B can be constructed from G in $\tilde{O}(p)$ time.

We call a 3-partition (L, S, R) of the vertices of B a *separator* if removing S from B disconnects L from R. We define the weight of the separator as the weight of S. Throughout this section, for a vertex $u \in V_B$, we use $N(u)$ to denote $\{v \in U_V \cup U_E \colon \{u, v\} \in E_B\}$ and $N[u]$ to denote $N(u) \cup \{u\}$. By the weights that we have assigned to vertices of B, we have the following observation:

Observation 2.1. Every min-weight separator (L, S, R) in B has $S \subseteq U_E$.

The following proposition relates min-cuts in G to minimum weight separators in B and translates our assumption that the min-cut $(C, V \setminus C)$ in G has $|C| \leq s$ into a constraint on a min-weight separator in B.

Proposition 2.3. The following hold:

1. If (L, S, R) is a min-weight separator in B, then $(\{v \colon u_v \in L\}, \{v \colon u_v \in R\})$ is a min-cut in G.

2. If $(C, V \setminus C)$ is a min-cut in G, then for

$$\begin{aligned}
L &:= \{u_v \colon v \in C\} \cup \{u_e \colon e \in E \text{ and } e \subseteq C\}, \\
S &:= \{u_e \colon e \in \delta(C)\}, \text{ and} \\
R &:= \{u_v \colon v \in V \setminus C\} \cup \{u_e \colon e \in E \text{ and } e \subseteq V \setminus C\},
\end{aligned}$$

the tuple (L, S, R) is a min-weight separator in B; moreover, if $|C| \leq s$, then $|L| \leq 3s^r$.

Moreover, an algorithm that runs in time $\tilde{O}(f(p, s, r, \lambda))$ to find a min-weight separator in B, for some function f, can be used to find a min-cut in G in time $\tilde{O}(p + f(p, s, r, \lambda))$.

Proof. Let (L, S, R) be a min-weight separator in B. By Observation 2.1 we have that $S \subseteq U_E$, and therefore $(\{v \colon u_v \in L\}, \{v \colon u_v \in R\})$ is a partition of V. Observation 2.1 also implies that $L \cap U_V$ and $R \cap U_V$ are both nonempty, and therefore $(\{v \colon u_v \in L\}, \{v \colon u_v \in R\})$ is a cut in G. Every hyperedge e crossing this cut must contain two vertices $x, y \in e$, where $u_x \in L$ and $u_y \in R$. For such a hyperedge e, the vertex u_e is adjacent to vertices in both L and R, and therefore, by the definition of a separator, we have that $u_e \in S$. Thus, we have that the weight of the cut $(\{v \colon u_v \in L\}, \{v \colon u_v \in R\})$ is at most $|S|$.

Next, let $(C, V \setminus C)$ be a min-cut in G, and let the sets L, S, R be as defined in the proposition. We note that, by the definition of B, removing S will disconnect L from R. Thus, (L, S, R) is indeed a separator. Furthermore, since $S \subseteq U_E$, the weight of (L, S, R) is $|S| = |\delta(C)|$.

The arguments in the previous two paragraphs imply that the min-cut capacity in G is at most the weight of a min-weight separator in B, and the weight of a minimum weight separator in B is at most the min-cut capacity in G. Thus, the min-cut capacity in G is equal to the weight of a min-weight separator in B.

Next, we show that if $|C| \leq s$, then $|L| \leq 3s^r$. We note that $L = \{u_v \colon v \in C\} \cup \{u_e \colon e \in E \text{ and } e \subseteq C\}$. We have that $|\{u_v \colon v \in C\}| \leq s$, and

$$|\{u_e \colon e \in E \text{ and } e \subseteq C\}| \leq \binom{s}{r} + \binom{s}{r-1} \dots \binom{s}{2} \leq s^r + \dots s^2 \leq 2s^r. \tag{2.64}$$

Thus, $|L| \leq s + 2s^r \leq 3s^r$.

The run-time to obtain a min-cut of G from a min-weight separator of B follows from the fact that B can be constructed in $\tilde{O}(p)$ time. QED.

By Proposition 2.3, in order to prove Theorem 2.13, it suffices to design an algorithm that runs in time $\tilde{O}_r(s^{3r}p)$ to find a min-weight separator (L, S, R) in the bipartite graph B, under the assumption that there exists a min-weight separator (L, S, R) in B with $|L| \leq 3s^r$. To achieve this, we use a slightly modified version of on algorithm of Li, Nanongkai, Panigrahi, Saranurak, and Yingchareonthawornchai [98] to obtain a minor of B, which we call a *kernel*, which has the following two properties with high probability:

1. A min-weight separator in the kernel can be used to find a min-weight separator in B in $\tilde{O}(p)$ time.

2. The kernel has at most $9rs^r$ vertices from U_V.

The first property helps in reducing the problem of finding a min-weight separator in B to the problem of finding a kernel and of finding a min-weight separator in the kernel. The second property allows us to design an algorithm which finds a min-weight separator in a kernel in time $\tilde{O}(\lambda r^2 s^{3r})$. In Algorithm 2.3, we give a randomized procedure that takes B, a set $X \subseteq V$ such that $X \cap L \neq \emptyset$, and an integer ℓ with $|L|/2 \leq \ell \leq |L|$ as inputs and returns a kernel. Running this algorithm $O(\log n)$ times with $X = U_V$ for each $\ell \in \{1, 2, 4, \ldots, 2^{\lceil \log(3s^r) \rceil}\}$ will return a kernel. We show the correctness of Algorithm 2.3 in Lemma 2.5. Unfortunately, the runtime of Algorithm 2.3 is $\tilde{O}(|E_H| \cdot |X|)$, which is too slow for our purposes. In Lemma 2.7, we show that there is a faster algorithm that returns the same kernel as Algorithm 2.3. Finally, in Lemma 2.8 we obtain an $\tilde{O}(s^{3r}p)$ time algorithm for finding a min-weight separator in B by combining the algorithm for finding a kernel with an algorithm due to Dinitz [110] which finds a min-weight separator in a kernel.

We will need the following definitions (we recall that the graph B has $m + n$ vertices):

Definition 2.3 ([98]). 1. For $X, Y \subseteq V_B$, we say that a separator (L, S, R) is an (X, Y)-*separator* if $X \subseteq L, Y \subseteq R$.

2. A minimum weight (X, Y)-separator will be denoted as a *min* (X, Y)*-separator*.

3. For a positive integer t, we say that a separator (L, S, R) is a t*-scratch* if $|S| \leq t$, $|L| \leq t/(100 \log(m + n))$, and $|S_{low}| \geq 300|L| \log(m + n)$, where $S_{low} := \{v \in S : \deg(v) \leq 8t\}$.

Definition 2.4. Let $x \in V_B$. We say that (B_x, Z_x, x, t_x) is a kernel if the following hold:

1. B_x is a subgraph of the graph obtained from B by contracting some subset $T_x \subseteq V_B \backslash \{x\}$ into the single vertex t_x such that B_x contains x and t_x.

2. Z_x is a subset of V_B such that (L, S, R) is a min $(\{x\}, \{t_x\})$-separator in B_x iff $(L', S \cup Z_x, R')$ is a min $(\{x\}, T_x)$-separator in B, where L' is the component of $V_B \setminus (S \cup Z_x)$ containing x, and $R' := V_B \setminus (L' \cup S \cup Z_x)$.

Let (L, S, R) be a min-weight separator in B, and let $x \in L$. The following lemma gives us a way to choose a set T_x so that with constant probability $\emptyset \neq T_x \subsetneq R$. For such a T_x, we have that (L, S, R) is a min $(\{x\}, T_x)$-separator, which means that we can potentially recover (L, S, R) from the min-weight separator in a kernel.

Lemma 2.4. Let (L, S, R) be a min-weight separator in B with $|L| \leq 3s^r$. Let $x \in L$ and let ℓ be a positive integer with $|L|/2 \leq \ell \leq |L|$. Let T be a subset of $V(B)$ chosen by sampling each vertex with probability $1/(8\ell)$, and let $T_x = T \setminus N[x]$. Then, we have that $\emptyset \neq T_x \subsetneq R$ with probability $\Omega(1)$.

Proof. By the hypothesis of Theorem 2.13, we have that $|S| = \lambda \geq 900s^r \log n$. We also have that $|L| \leq 3s^r \leq |S|/2$. We also note that a random separator consisting of the neighbors of a single vertex of U_V has expected size at most $r|U_E|/|U_V|$, so $|S| \leq (r/n)|U_E| \leq |U_E|/2$. Consequently, we have that

$$|(U_V \cup U_E) \setminus N[x]| \geq |U_E| - |S| - |L| \geq |U_E| - (|U_E|/2) - (|U_E|/4) \geq |U_E|/4 \geq |S|/4 \geq s^r. \tag{2.65}$$

Since each vertex of $V(B) \setminus N[x]$ is included in T_x with probability $1/(8\ell) \geq 1/24s^r$, it follows that at least one of them will be included in T_x with at least constant probability.

Now we note that since $N(x)$ is a separator in B, $|N(x)| \geq \lambda$. Thus, $|(L \cup S) \setminus N[x]| \leq (|L| + \lambda) - \lambda = |L|$. Since each of these vertices is included in T_x with probability at most $1/(8(|L|/2)) = 1/(4|L|)$, the probability that none of them are included is at least a constant.

Since $|R| \geq 1$ by the definition of a separator, and since each vertex of R is included in T_x with probability $1/(8\ell) \leq 1/8$, the probability that at least one vertex of R is excluded from T_x is also at least a constant. QED.

Our technique to find a min-weight separator in B is to find, for each $x \in U_V$, a kernel with $O(\lambda r s^{2r})$ edges. Algorithm 2.3 is a slightly modified version of an algorithm of Li, Nanongkai, Panigrahi, Saranurak, and Yingchareonthawornchai [98] for computing a small kernel. The difference is that while our algorithm contracts $T'_x := T_x \cup (N(T_x) \cap U_V)$, their algorithm merely contracts T_x. We need this additional contraction in order to prove Lemma 2.6, which bounds the size of the kernel. This bound on the size of the kernel that we obtain is necessary to prove Claim 2.9, which shows that we can find a min-weight separator in a kernel efficiently. Additionally, while Li, Nanongkai, Panigrahi, Saranurak, and Yingchareonthawornchai [98] analyzed the algorithm only for an unweighted graph, we run it on the graph B, which has vertex weights (of a special kind).

Now we show the correctness of Algorithm 2.3.

Lemma 2.5. Consider executing Algorithm 2.3 on input (B, X, ℓ) for some positive integer ℓ such that $|L|/2 \leq \ell \leq |L|$, where X is a subset of U_V and (L, S, R) is a min-weight separator in B. For $x \in X \cap L$, if the set T_x chosen by the algorithm satisfies $\emptyset \neq T_x \subsetneq R$, then the tuple (B_x, Z_x, x, t_x) returned by the algorithm is a kernel.

```
FindKernel($B, X, \ell$):
  $T \leftarrow \emptyset$
  For $v$ in $V(B)$
      Add $v$ to the set $T$ with probability $1/(8\ell)$
  For $x \in X$
      $T_x \leftarrow T \setminus N[x]$
      $T'_x \leftarrow T_x \cup (N(T_x) \cap U_V)$
      Obtain $B'$ from $B$ by contracting the vertices of $T'_x$ into a single vertex $t_x$
      $Z_x \leftarrow \{v \in V(B') \colon \{x, v\}, \{v, t_x\} \in E(B')\}$
      Obtain $B''$ from $B'$ by deleting the vertices of $Z_x$ and all edges fully contained in $N(t_x)$
      Obtain $B'''$ from $B''$ by deleting all vertices that are disconnected from $x$ in $B'' \setminus N[t_x]$
      Obtain $B_x$ from $B'''$ by deleting from $B'''$ all neighbors of $t_x$ with degree one
  Return $\{(B_x, Z_x, x, t_x) \colon x \in X\}$
```

Algorithm 2.3: FindKernel

Proof. Suppose that $\emptyset \neq T_x \subsetneq R$. Then, (L, S, R) is a min $(\{x\}, T_x)$-separator, and contracting T_x cannot destroy (L, S, R) (or any other min $(\{x\}, T_x)$-separator). Let R' be R with the vertices of $T'_x = T_x \cup (N(T_x) \cap U_V)$ contracted into the single vertex t_x. By Observation 2.1 we have that $S \subseteq U_E$, which means that since $T_x \subseteq R$ then $N(T_x) \cap U_V \subseteq R$ as well. Thus, (L, S, R') is a min-weight separator in B'. Therefore, $(B', \emptyset, x, t_x)$ is a kernel.

By Observation 2.1, we have that the weight of the separator (L, S, R) is $|S|$. Consequently, we have the following observation:

Observation 2.2. There exist $|S|$ vertex-disjoint $x - t_x$ paths in B', each of which visits exactly one vertex from $N(x)$ and exactly one vertex from $N(t_x)$.

Since every edge in $B'[N(x)]$ or $B'[N(t_x)]$ is not present in any of these paths, removing all of these edges does not affect the weight of a min $(\{x\}, \{t_x\})$-separator in B'. On the other hand, every vertex $v \in N(x) \cap N(t_x)$ is included in every $(\{x\}, \{t_x\})$-separator in B'. Therefore, (B'', Z_x, x, t_x) is a kernel.

By Observation 2.2, removing from B'' vertices which are disconnected from x in $B'' \setminus N[t_x]$ cannot affect the size of a min $(\{x\}, \{t_x\})$-separator, since such vertices cannot participate in any $s - t_x$ paths which use only a single vertex of $N(t_x)$. Thus, (B''', Z_x, x, t_x) is also a kernel.

Finally, we note that degree one neighbors of t_x cannot participate in any $x - t_x$ paths using only one vertex from $B'''[N[t_x]]$. Thus, again by Observation 2.2, removing such vertices does not affect the weight of a min $(\{x\}, \{t_x\})$-separator, and thus (B_x, Z_x, x, t_x) is a kernel. QED.

By Lemmas 2.4 and 2.5, running Algorithm 2.3 $O(\log n)$ times with $X = U_V$ for each

$\ell \in \{1, 2, 4, 8, ..., 2^{\lceil \log(3s^r) \rceil}\}$ will return a kernel (B_x, Z_x, x, t_x) for some $x \in L$ with high probability, where (L, S, R) is a min-weight separator of B with $|L| \leq 3s^r$. We next show that the kernel computed by Algorithm 2.3 is not too large.

Lemma 2.6. Consider executing Algorithm 2.3 on input (B, X, ℓ) for some positive integer ℓ such that $|L|/2 \leq \ell \leq |L|$, where X is a subset of U_V and (L, S, R) is a min-weight separator in B with $|L| \leq 3s^r$. For $x \in X \cap L$, if the set T_x chosen by the algorithm satisfies $\emptyset \neq T_x \subsetneq R$ and the tuple (B_x, Z_x, x, t_x) returned by the algorithm is a kernel, then B_x contains at most $9rs^r$ vertices from U_V with high probability.

Proof. Consider (L', S', R') where $S' := S \setminus Z_x$, L' is the component of $V[B_x] \setminus S'$ which contains x, and $R' := V[B_x] \setminus (L' \cup S')$. Since $T_x \subseteq R$ and (B_x, Z_x, x, t_x) is a kernel, we have that (L', S', R') is a min-weight (x, t_x) separator. We know that $|L' \cap U_V| \leq |L| \leq 3s^r \leq rs^r$ and that $S' \subseteq U_E$, so it suffices to show that $|R' \cap U_V| \leq 8rs^r$.

Consider an arbitrary vertex $v \in R' \cap U_V$. Since v was not contracted into t_x, it must be the case that v was not in T_x, and none of v's neighbors were in T_x. If v had more than $8\ell \log n$ neighbors in $V(B) \setminus N(x)$, then with high probability, one of the neighbors of v would be in T_x. Therefore, with high probability, v has at most $8\ell \log n$ neighbors in $V(B) \setminus N(x)$. We know, however, that v has at least λ neighbors in B. Therefore, v has at least $\lambda - 8\ell \log n$ neighbors in $N(x)$.

Since $x \in L$, we have that $|N(x)| \leq |L| + \lambda \leq 3s^r + \lambda$. Since $x \in U_V$, we also have that $N(x) \subseteq U_E$. Since G has rank r, it follows that each vertex in $N(x)$ has degree at most r. Thus, the vertices of $N(x)$ collectively have at most $r(3s^r + \lambda)$ edges incident to them. Since each vertex of $R' \cap U_V$ is incident to at least $\lambda - 8\ell \log n$ of those edges, we have that

$$|R' \cap U_V| \leq \frac{r(3s^r + \lambda)}{\lambda - 8\ell \log n}. \tag{2.66}$$

From the hypothesis of Theorem 2.13, we have that $\lambda \geq 900s^r \log n$. From the hypothesis of the claim we are proving, we have that $\ell \leq 3s^r$. Therefore, $\lambda - 8\ell \log n \geq \lambda/2$, so

$$\frac{r(3s^r + \lambda)}{\lambda - 8\ell \log n} \leq \frac{2r(3s^r + \lambda)}{\lambda} \leq 2r(3s^r + 1) \leq 8rs^r. \tag{2.67}$$

QED.

The following lemma will be useful in showing that the kernel generated by Algorithm 2.3 can be obtained efficiently.

Lemma 2.7. There is a randomized algorithm that takes as input the bipartite graph B, a subset $X \subseteq U_V$, and positive integers ℓ, t, chooses a subset T of $V(B)$ by sampling each vertex with probability $1/(8\ell)$, and runs in time $\tilde{O}(|E(H)| + |X|t\ell)$ to return a tuple (B_x, Z_x, x, t_x) for every $x \in X$. Additionally, the returned collection satisfies the following with high probability:

- For $x \in X$, if B contains a t-scratch (L, S, R) with $x \in L$, $|L|/2 \leq \ell \leq |L|$, and $\emptyset \neq T \setminus N[x] \subseteq R$, then the tuple (B_x, Z_x, x, t_x) returned by the algorithm is a kernel with $|E(B_x)| = O(t\ell \log p)$ and B_x contains at most $9rs^r$ vertices from U_V.

Proof. This follows directly from the results of [98]. They design an algorithm (Algorithm 1 in [98]) which chooses a set T according to the same distribution as Algorithm 2.3, but uses linear sketching and BFS-like computations starting from x to compute a tuple (B_x, Z_x, x, t_x) more efficiently than Algorithm 2.3; moreover, if B contains a t-scratch (L, S, R) with $x \in L$, $|L|/2 \leq \ell \leq |L|$, and $\emptyset \neq T \setminus N[x] \subseteq R$, then with high probability the tuple (B_x, Z_x, x, t_x) returned by their algorithm is a kernel with $|E(B_x)| = O(t\ell \log p)$, and this kernel is the tuple that would be output by their version of Algorithm 2.3 when run on the same input with the same source of randomness.

We obtain an algorithm with the guarantees given in the statement of the lemma by applying the modifications described in [98] to our version of Algorithm 2.3. There are only two differences between Algorithm 2.3 and the corresponding algorithm in [98]. First, their algorithm is only for unweighted graphs, while we run Algorithm 2.3 on the weighted graph B. Second, we contract $T'_x := T_x \cup (N(T_x) \cap U_V)$ rather than just T_x. We now show that neither of those differences affects the correctness or runtime of the modified algorithm.

We note that since Algorithm 2.3 does not even look at the weights on the vertices, the fact that B is weighted does not affect the runtime of our algorithm.

The additional contraction of $N(T_x) \cap U_V$ into t_x can be performed without increasing the runtime of the algorithm, and as shown in Lemma 2.5 the resulting algorithm still finds a kernel whenever $\emptyset \neq T_x \subsetneq R$. Furthermore, by Lemma 2.6, this kernel contains at most $9rs^r$ vertices from U_V with high probability. We skip the detailed description and analysis of the faster algorithm for finding kernels, since it is identical to [98]. QED.

The following lemma completes the proof of Theorem 2.13.

Lemma 2.8. There exists a randomized algorithm that runs in time $\tilde{O}(s^{3r}p)$ to find a min-weight separator in B with high probability.

Proof. Let k be the approximation of λ obtained by running the algorithm from Theorem 2.6. To find a min-weight separator in B, we run the algorithm from Lemma 2.7 on B with $t = k + r + 300s^r \log n$, and $X = U_V$ for $\Theta(\log n)$ times for each $\ell \in \{1, 2, 4, 8, \dots, 2^{\lceil \log(3s^r) \rceil}\}$.

Then, for each tuple (B_x, Z_x, x, t_x) returned, if B_x contains at most $9rs^r$ vertices from U_V then we compute a min $(\{x\}, \{t_x\})$-separator in B_x. For each such separator (L, S, R), we compute a corresponding separator (L', S', R') in B by setting $S' := S \cup Z_x$, L' to be the component of $V_B \setminus S'$ which contains x, and $R' := V_B \setminus (L' \cup S')$. By Definition 2.4, if (B_x, Z_x, x, t_x) is a kernel then (L', S', R') is a min-weight $(\{x\}, T_x)$-separator in B. We return the cheapest among all computed separators.

We first analyze the correctness of our procedure. We will need the following claim:

Claim 2.8. Let (L, S, R) be a min-weight separator in B with $|L| \leq 3s^r$. Then, (L, S, R) is a t-scratch.

Proof. Since each min-weight separator in B corresponds to a min-cut in G, we know that $|S| = \lambda \leq k \leq t$. By assumption and by our choice of t, we have that $|L| \leq 3s^r \leq t/(100 \log n)$. By Observation 2.1, we have that $S \subseteq U_E$. Also, by the definition of B, every vertex in U_E has degree at most r, which is less than $8t$. Therefore, we have that $S_{low} = S$. Thus, $|S_{low}| = |S| = \lambda \geq 900s^r \log n \geq 300|L| \log n$, so (L, S, R) is a t-scratch. QED.

Let (L, S, R) be a min-weight separator in B with $|L| \leq 3s^r$. By Claim 2.8, (L, S, R) is a t-scratch. Since we run the algorithm of Lemma 2.7 $\Theta(\log n)$ times for each $\ell = 1, 2, 4, 8, \ldots, 2^{\lceil \log(3s^r) \rceil}$, we will run the algorithm $\Theta(\log n)$ times for some $\ell \in [|L|/2, |L|]$. By Lemma 2.4, for each $x \in L$, each time we run the algorithm, the set T that is chosen satisfies $\emptyset \neq T_x \subsetneq R$ with constant probability, so with high probability this occurs for some $x \in L$ at least once. Lemma 2.7 tells us that when the algorithm chooses such a T, then with high probability it will return a kernel (B_x, Z_x, x, t_x) where B_x has at most $9rs^r$ vertices from U_V. By Definition 2.4, a min $(\{x\}, \{t_x\})$-separator (L, S, R) in this kernel gives a min $(\{x\}, T_x)$-separator $(L', S \cup Z_x, R')$ in B, where L' is the component of $V_B \setminus (S \cup Z_x)$ containing x, and $R' = V_B \setminus (L' \cup S \cup Z_x)$. Since $T_x \subseteq R$, a min $(\{x\}, T_x)$-separator in B is in fact a min-weight separator in B. Thus, the algorithm finds a min-weight separator in B with high probability.

We now analyze the runtime of our procedure. The algorithm from Lemma 2.7 runs in time $\tilde{O}(|E_B| + |U_V|(k + r + 300s^r)(3s^r)) = \tilde{O}(p + n(\lambda + r + s^r)s^r) = \tilde{O}(p + ps^r + rns^{2r}) = \tilde{O}(ps^r + rns^{2r})$. Running the algorithm $\Theta(\log n)$ times each for $\log(3s^r)$ different values of ℓ increases the running time by a factor of $O(\log(3s^r) \log n) = O(r(\log s)(\log n)) = O(r \log^2 n)$ giving us a total runtime of $\tilde{O}(prs^r + r^2ns^{2r})$. To bound the time to compute a min $(\{x\}, \{t_x\})$-separator in each kernel we use the following claim.

Claim 2.9. If (B_x, Z_x, x, t_x) is a tuple returned by the algorithm from Lemma 2.7 with at most $9rs^r$ vertices from U_V and $O(\lambda rs^{2r})$ edges, then a min $(\{x\}, \{t_x\})$-separator in B_x can be computed in time $\tilde{O}(\lambda r^2 s^{3r})$.

Proof. Since B is bipartite, every path of length d in B visits at least $\lceil d/2 \rceil$ vertices in U_V. The graph B_x is obtained from a bipartite graph by contracting some subset of vertices. Therefore, except for the vertex t_x created by the contraction, every vertex in B_x which is not in U_V must have all of its neighbors in U_V. This means that for every path in d, every pair of consecutive vertices in the path not including the vertex generated by the contraction has at least one vertex from U_V. Therefore, every path of length d in B_x visits at least $\lceil (d-2)/2 \rceil$ vertices from U_V. By the assumption of the claim, B_x contains $O(rs^r)$ vertices from U_V. Thus, every path in B_x has length $O(rs^r)$.

Therefore, we can use Dinitz's algorithm to efficiently find a min $(\{x\}, \{t_x\})$-separator in B_x [110]. Since the lengths of paths in B_x are bounded by $O(rs^r)$, the number of blocking flows found by Dinitz's algorithm is also $O(rs^r)$. Each blocking flow can be found in time proportional to the number of edges in B_x, and by assumption this is $\tilde{O}(\lambda rs^{2r})$. Therefore, the total runtime of Dinitz's algorithm on a B_x is $\tilde{O}(\lambda r^2 s^{3r})$. QED.

Let (B_x, Z_x, x, t_x) be a kernel returned by the algorithm from Lemma 2.7. By Lemma 2.7, we have that B_x contains at most $9rs^r$ vertices from U_V and $O(\lambda rs^{sr})$ edges with high probability. By Claim 2.9, we can compute a min-weight separator for a single B_x with at most $9rs^r$ vertices from U_V and $O(\lambda rs^{sr})$ edges in time $\tilde{O}(\lambda r^2 s^{3r})$. Thus, we can compute a min-weight separator for all B_x with at most $9rs^r$ vertices from U_V and $O(\lambda rs^{sr})$ edges in time $\tilde{O}_r(s^{3r}\lambda n) = \tilde{O}_r(s^{3r}p)$.

QED.

2.4.4 Expander Decomposition Based Min-Cut

In this section, we give an algorithm to find connectivity in hypergraphs in which every min-cut has large size. In order to analyze the run-time of our algorithm, we need to bound the run-time to implement Trim and Shave operations. This is summarized in the following claim.

Claim 2.10. Let $G = (V, E)$ be an n-vertex r-rank hypergraph of size p with $p \geq n$, and let $\mathcal{X} = \{X_1, \ldots, X_k\}$ be a collection of disjoint subsets of V. Then, there exists a deterministic algorithm which takes as input G and $\mathcal{X}$, and runs in time $O(pr)$ to return $\{\textsc{Trim}(X_1), \ldots, \textsc{Trim}(X_k)\}$ and $\{\textsc{Shave}(X_1), \ldots, \textsc{Shave}(X_k)\}$.

Proof. For $v \in V$, let $part(v) := i$ if $v \in X_i$ and 0 otherwise. For $e \in E$, let $epart(e) := i$ if $e \subseteq X_i$ and 0 otherwise. Let $d_{\mathcal{X}}(v) := d_{X_i}(v)$ if $v \in X_i$ and 0 otherwise and $\delta_{\mathcal{X}}(v) := \delta_{X_i}(v)$

if $v \in X_i$ and $\emptyset$ otherwise. We also recall that the degree of a vertex in G is denoted by $d(v)$. The functions $part : V \to \{0, \ldots, k\}$, $d : V \to \mathbb{Z}_+$, $d_{\mathcal{X}} : V \to \mathbb{Z}_{\geq 0}$, and $\delta_{\mathcal{X}} : V \to 2^E$ can be computed in $O(p)$ time using Algorithm 2.4.

```
TrimShaveHelper($G = (V, E), \mathcal{X} = \{X_1, \ldots, X_k\}$):
  For all $v \in V$
      $part(v), d(v), d_{\mathcal{X}}(v) \leftarrow 0$
      $\delta_{\mathcal{X}}(v) \leftarrow \emptyset$
  For $i = 1, \ldots, k$
      For all $v \in X_i$
          $part(v) \leftarrow i$
  For all $e \in E$
      $v \leftarrow$ an arbitrary vertex of $e$
      $epart(e) \leftarrow part(v)$
      For all $u \in e \setminus \{v\}$
          If $part(u) \neq part(v)$
              $epart(e) \leftarrow 0$
      If $epart(e) \neq 0$
          For all $v \in e$
              $d_{\mathcal{X}}(v) \leftarrow d_{\mathcal{X}}(v) + 1$
              Add $e$ to $\delta_{\mathcal{X}}(v)$
  Return $part, d, d_X, \delta_X$
```

Algorithm 2.4: TrimShaveHelper

We claim that Algorithm 2.4 can be implemented to run in time $O(p)$. We note that looping over all of the vertices takes time $O(n)$. Since the sets X_is are disjoint, looping over all the vertices of each X_i also takes time $O(n)$. Looping over all edges and within each edge looping over all vertices in the edge takes time $O(\sum_{e \in E} |e|) = O(p)$. Since $n \leq p$, the complete algorithm can be implemented to run in time $O(p)$.

We use Algorithm 2.5 to perform the Shave operation. Since the sets X_is are disjoint, iterating over all vertices of each X_i takes time $O(n)$, so the portion of the algorithm outside of the call to TrimShaveHelper can be implemented to run in time $O(p)$, and therefore the whole algorithm can be as well.

We use Algorithm 2.6 to perform the Trim operation. Here the set F stores vertices that the algorithm has yet to examine, and T stores vertices that the algorithm has determined must be trimmed. The values of $d_{\mathcal{X}}(v)$ and $\delta_{\mathcal{X}}(v)$ are updated as vertices are trimmed, and whenever $d_{\mathcal{X}}(v)$ changes for a vertex v, the algorithm verifies whether that vertex v should be trimmed. Removing vertices from X_i cannot increase $d_{X_i}(v)$ for any vertex $v \in X_i$. Therefore, if a vertex v satisfies $d_{X_i}(v) < d(v)/2r$ when v is added to T, v will still satisfy this inequality when it is actually removed from X_i. We note that each vertex is chosen at

```
ShaveAlgorithm($G = (V, E), \mathcal{X} = \{X_1, \ldots, X_k\}$):
    $part, d, d_{\mathcal{X}}, \delta_{\mathcal{X}} \leftarrow$ TrimShaveHelper($G, \mathcal{X}$)
    For $i = 1, \ldots, k$
        For all $v \in X_i$
            If $d_{\mathcal{X}}(v) \leq (1 - 1/r^2)d(v)$
                Remove $v$ from $X_i$
    Return $\mathcal{X}$
```

Algorithm 2.5: ShaveAlgorithm

most once as the arbitrary vertex from F, since whenever a vertex is chosen from F it is removed from F and never added back to it. Whenever a vertex v is chosen from T, it is removed from T, and $part(v)$ is set to 0. Since the algorithm only adds a vertex v to T only if $part(v) \neq 0$, each vertex is chosen at most once as the arbitrary vertex from T.

The case of choosing a vertex from F can be implemented to run in constant time since we perform an arithmetic comparison and a constant number of set additions and removals. When the algorithm selects a vertex from T, it iterates over all hyperedges of that vertex which are currently in $X_{part(v)}$. For each of these hyperedges e, all iterations of the innermost **for** loop can be implemented to run in time $O(|e|)$. Thus, the total time spent processing a vertex v from T is $O(\sum_{e\in\delta(v)} |e|)$. Since each vertex is processed in T at most once, the total time to process all of the vertices is $O(\sum_{v\in V} \sum_{e\in\delta(v)} |e|) = O(pr)$. QED.

The following theorem is the main result of this section.

Theorem 2.14. Let $G = (V, E)$ be an n-vertex r-rank simple hypergraph of size p with connectivity λ such that $\lambda \geq r(4r^2)^r$ and every min-cut $(C, V \setminus C)$ in G has $\min\{|C|, |V \setminus C|\} > r - \log(\lambda/4r)/\log n$. Then, there exists a randomized algorithm which takes G as input and runs in time

$$\tilde{O}_r\left(p + (\lambda n)^{1+o(1)} + \min\left\{\lambda^{\frac{r-3}{r-1}+o(1)}n^{2+o(1)}, \frac{n^{r+r\cdot o(1)}}{\lambda^{\frac{r}{r-1}}}, \lambda^{\frac{5r-7}{4r-4}+o(1)}n^{\frac{7}{4}+o(1)}\right\}\right),$$

to return a min-cut of G with high probability.

Proof. We will use algorithms from Theorems 2.6, 2.7, 2.8, and 2.9, Corollary 2.1, and Lemma 2.3 to find a min-cut. We use CXApproximation to find a constant-approximation k for the min-cut capacity. Next, we use Certificate(G, k) to get a graph G' with size $p_1 = O(rkn) = O(r\lambda n)$. Next, we find an expander decomposition $\mathcal{X}$ of G' with respect to the parameter $\phi := (6r^2/\delta)^{1/(r-1)}$. We then apply the Trim operation on $\mathcal{X}$ followed by $3r^2$ Shave operations to obtain a collection $\mathcal{X}''$ of disjoint subsets of V. We contract all

```
TrimAlgorithm($G = (V, E), \mathcal{X} = \{X_1, \ldots, X_k\}$):
    $x, y, d, d_{\mathcal{X}}, \delta_{\mathcal{X}} \leftarrow$ TrimShaveHelper($G, \mathcal{X}$)
    $T \leftarrow \emptyset$
    $F \leftarrow V$
    While $T \cup F \neq \emptyset$
        If $T \neq \emptyset$
            Let $v \in T$ be arbitrary
            Remove $v$ from $T$ and from $X_{part(v)}$
            $part(v) \leftarrow 0$
            For all $e \in \delta_{\mathcal{X}}(v)$
                For $u \in e \setminus \{v\}$
                    Remove $e$ from $\delta_{\mathcal{X}}(u)$
                    $d_{\mathcal{X}}(u) \leftarrow d_{\mathcal{X}}(u) - 1$
                    If $d_{\mathcal{X}}(u) < d(u)/2r$ and $u \notin T$
                        Add $u$ to $T$
            Else
                Let $v \in F$ be arbitrary
                remove $v$ from $F$
                If $d_{\mathcal{X}}(v) < d(v)/2r$ and $part(v) \neq 0$
                    Add $v$ to $T$
        Return $\mathcal{X}$
```

Algorithm 2.6: TrimAlgorithm

these subsets and run either FPZMinCut or CXMinCut or CQMincut on the resulting hypergraph to find a min-cut. We present the pseudocode of our approach in Algorithm 2.7.

We now prove the correctness of ExpDecompMinCut. By hypothesis, we know that every min-cut $(C, V \setminus C)$ has $\min\{|C|, |V \setminus C|\} > r - \log(\lambda/4r)/\log n$. Hence, by Lemma 2.1, every min-cut $(C, V \setminus C)$ has $\min\{|C|, |V \setminus C|\} > (\lambda/2)^{1/r} \geq 4r^2$ (by assumption on λ).

By Theorem 2.6, we have that $k > \lambda$. Therefore, by Theorem 2.7, the subhypergraph G' is a simple hypergraph with connectivity λ and the min-cuts in G' are exactly the min-cuts of G. Consequently, the min-degree δ' in G' is at least the connectivity λ. Let $(C, V \setminus C)$ be an arbitrary min-cut in G' and let $\mathcal{X} = (X_1, \ldots, X_t)$ be the expander decomposition of G' for the parameter ϕ. By definition, $\phi \leq 1/(r-1)$. Hence, by Lemma 2.3, for every $i \in [t]$, we have that

$$\delta' \geq \lambda \geq |E^o_{G'}(X_i \cap C, X_i \cap (V \setminus C))| \tag{2.68}$$

$$\geq \min\left\{\left(\frac{6r^2}{\delta'}\right)^{\frac{1}{r-1}}, \frac{1}{r-1}\right\} \min\{\mathrm{vol}_{G'}(X_i \cap C), \mathrm{vol}_{G'}(X_i \setminus C)\} \tag{2.69}$$

$$\geq \min\left\{\left(\frac{6r^2}{\delta'}\right)^{\frac{1}{r-1}}, \frac{1}{r-1}\right\} \delta' \min\{|X_i \cap C|, |X_i \setminus C|\}. \tag{2.70}$$

```
ExpDecompMinCut($G$):
    $k \leftarrow$ CXApproximation($G$)
    $G' \leftarrow$ Certificate($G, k$)
    $\delta' \leftarrow \min_{v \in V} d_{G'}(v)$
    $\phi \leftarrow \min\{(6r^2/\delta')^{1/(r-1)}, 1/(r-1)\}$
    $\mathcal{X} \leftarrow$ ExpanderDecomposition($G', \phi$)
    $\mathcal{X}' \leftarrow$ TrimAlgorithm($G', \mathcal{X}$)
    $\mathcal{X}'' \leftarrow \mathcal{X}'$
    For $i = 1, \ldots, 3r^2$
        $\mathcal{X}'' \leftarrow$ ShaveAlgorithm($\mathcal{X}''$)
    Let $G''$ be the hypergraph obtained by contracting each set $X'' \in \mathcal{X}''$ into a single vertex
    Run CXMinCut($G''$), FPZMinCut($G''$), and CQMinCut($G''$) simultaneously,
        return the output of the earliest terminating among the three, and terminate
```

Algorithm 2.7: ExpDecompMinCut

Thus, $\min\{|X_i \cap C|, |X_i \setminus C|\} \leq \max\{(\delta'/6r^2)^{1/(r-1)}, r-1\}$ for every $i \in [t]$.
If $\max\{(\delta'/6r^2)^{1/(r-1)}, r-1\} = r-1$, then $\min\{|X'_i \cap C|, |X'_i \setminus C|\} \leq r-1 \leq 3r^2$ for every $i \in [t]$ and if $\max\{(\delta'/6r^2)^{1/(r-1)}, r-1\} = (\delta'/6r^2)^{1/(r-1)}$, then by Claim 2.1, we have that $\min\{|X'_i \cap C|, |X'_i \setminus C|\} \leq 3r^2$ for every $i \in [t]$. We recall that $\lambda \geq r(4r^2)^r$ and every min-cut has size at least $4r^2$. Hence, by $3r^2$ repeated applications of Claim 2.2, we have that $\min\{|X''_i \cap C|, |X''_i \cap (V \setminus C)|\} = 0$ for every $i \in [t]$. In particular, this means that contracting each X''_i does not destroy $(C, V \setminus C)$, so $\delta(C)$ is still a min-cut set in G''. Since contraction cannot decrease the capacity of any cut, this implies that every min-cut in G'' is a min-cut in G', and thus, Algorithm 2.7 returns a min-cut.

We now bound the run-time of Algorithm 2.7. By Theorem 2.6, CXApproximation runs in time $O(p)$ to return $k \leq 3\lambda$. By Theorem 2.7, Certificate also runs in time $O(p)$ to return a hypergraph $G' = (V, E')$ of size $p_1 := \sum_{e \in E'} |e| = O(rkn) = O(r\lambda n)$ and having min-degree $\delta' \geq \lambda$. By Lemma 2.3, we obtain an expander decomposition $\mathcal{X}$ of G' in time $O\left(p_1^{1+o(1)}\right)$ such that the number of hyperedges of G' intersecting multiple parts of $\mathcal{X}$ is $O(r\phi p_1^{1+o(1)})$. By Claim 2.10, we can implement a Trim operation followed by $3r^2$ Shave operations to obtain a collection $\mathcal{X}''$ of disjoint subsets of V in time $O(p_1 r \cdot 3r^2) = O(p_1 r^3)$. By Claim 2.3 and $3r^2$ repeated applications of Claim 2.4 (similar to the proof of Theorem 2.2), we have that $|E''| = O(4r^{9r^2+1}\phi p_1^{1+o(1)})$. Since contraction cannot decrease the degree of any vertex, every vertex in $G'' = (V'', E'')$ has degree at least δ'. Therefore, the number of vertices n_2 in G'' satisfies $n_2\delta' \leq \sum_{v \in V''} d_{G''}(v) = \sum_{e \in E''} |e| \leq r|E''|$, which implies that

$$n_2 = O\left(\frac{4r^{9r^2+2}\phi p_1^{1+o(1)}}{\delta'}\right) = O\left(\frac{4r^{9r^2+2}\phi p_1^{1+o(1)}}{\lambda}\right). \tag{2.71}$$

Let $p_2 := \sum_{e \in E''} |e|$. Since contraction cannot increase the size of the hypergraph, we have that $p_2 \leq p_1$. Moreover, contraction does not increase the rank and hence, the rank of G'' is at most r. Therefore, the run-time of CXMinCut on G'' is

$$O(p_2 + r\lambda n_2^2) = O\left(p_1 + \frac{16r^{18r^2+4}\phi^2 p_1^{2+o(1)}}{\lambda}\right) \tag{2.72}$$

$$= O\left(r\lambda n + \frac{r^{19r^2}(\lambda n)^{2+o(1)}\phi^2}{\lambda}\right) \tag{2.73}$$

$$= O\left(r\lambda n + r^{19r^2}\lambda^{1+o(1)}n^{2+o(1)}\phi^2\right) \tag{2.74}$$

$$= O\left(r\lambda n + r^{20r^2}\lambda^{\frac{r-3}{r-1}+o(1)}n^{2+o(1)}\right). \tag{2.75}$$

The second equation is by the bound on p_1, and the last equation is by the setting of ϕ and the fact that $\delta' \geq \lambda$.

The run-time of FPZMinCut on G'' is

$$\tilde{O}\left(p_2 + n_2^r\right) = \tilde{O}\left(p_1 + \left(\frac{4r^{9r^2+2}\phi p_1^{1+o(1)}}{\lambda}\right)^r\right) \tag{2.76}$$

$$= \tilde{O}\left(\lambda n + \frac{r^{11r^3}(\lambda n)^{r(1+o(1)}\phi^r}{\lambda^r}\right) \tag{2.77}$$

$$= \tilde{O}\left(\lambda n + r^{11r^3}\lambda^{r\cdot o(1)}n^{r(1+o(1))}\phi^r\right) \tag{2.78}$$

$$= \tilde{O}\left(\lambda n + \frac{r^{15r^3}n^{r(1+o(1))}}{\lambda^{\frac{r}{r-1}-r\cdot o(1)}}\right). \tag{2.79}$$

Again, the second equation is by the bound on p_1, and the last equation is by the setting of ϕ and using the fact that $\delta' \geq \lambda$.

Let m_2 be the number of edges in G''. We assume that G'' is connected (otherwise the problem is trivial). Thus we have that $n_2 \leq p_2$, and therefore $(m_2 + n_2) \leq 2p_2$. Since $p_2 \leq rm_2$, this bound is tight, up to a factor of r. Therefore, the run-time of CQMinCut on G'' is

$$\begin{aligned}
\tilde{O}\left(\sqrt{p_2 n_2 (m_2+n_2)^{1.5}}\right) &= \tilde{O}\left(\sqrt{p_2 n_2 (2p_2)^{1.5}}\right) && (2.80)\\
&= \tilde{O}\left(\sqrt{p_1^{2.5} n_2}\right) && (2.81)\\
&= \tilde{O}\left(p_1^{1.25} n_2^{0.5}\right) && (2.82)\\
&= \tilde{O}\left((\lambda n)^{1.25}\left(\frac{4r^{9r^2+2}\phi(\lambda n)^{1+o(1)}}{\lambda}\right)^{0.5}\right) && (2.83)\\
&= \tilde{O}\left(r^{4.5r^2+1}\lambda^{1.25+o(1)} n^{1.75+o(1)}\left(\frac{6r^2}{\lambda}\right)^{\frac{1}{2(r-1)}}\right) && (2.84)\\
&= \tilde{O}\left(r^{4.5r^2+2}\lambda^{\frac{5r-7}{4r-4}+o(1)} n^{1.75+o(1)}\right). && (2.85)
\end{aligned}$$

The fourth equation is by the bound on p_1, and the second-to-last equation is by the setting of ϕ and the fact that $\delta' > \lambda$.

Since the algorithm runs the fastest algorithm among CXMinCut, FPZMinCut, and CQMinCut, the overall runtime is

$$\tilde{O}\Bigg(p + p_1^{1+o(1)} + r^3 p_1 + \min\left\{r\lambda n + r^{20r^2}\lambda^{\frac{r-3}{r-1}+o(1)} n^{2+o(1)}, \lambda n + \frac{r^{15r^3} n^{r(1+o(1))}}{\lambda^{\frac{r}{r-1}-r\cdot o(1)}}, r^{4.5r^2+2}\lambda^{\frac{5r-7}{4r-4}+o(1)} n^{\frac{7}{4}+o(1)}\right\}\Bigg) \quad (2.86)$$

$$= \tilde{O}\Bigg(p + (r\lambda n)^{1+o(1)} + r^3\lambda n + \min\left\{r^{20r^2}\lambda^{\frac{r-3}{r-1}+o(1)} n^{2+o(1)}, \frac{r^{15r^3} n^{r(1+o(1))}}{\lambda^{\frac{r}{r-1}-r\cdot o(1)}}, r^{4.5r^2+2}\lambda^{\frac{5r-7}{4r-4}+o(1)} n^{\frac{7}{4}+o(1)}\right\}\Bigg). \quad (2.87)$$

The run-time bound stated in the theorem follows by observing that $\lambda n \le \delta n \le p$, where δ is the min-degree of the input graph. QED.

Remark 2.3. We can obtain a deterministic counterpart of Theorem 2.14 by modifying the last step of Algorithm 2.7 to only run CXMinCut. The resulting algorithm has a run-time of

$$\tilde{O}_r\left(p + (\lambda n)^{1+o(1)} + \lambda^{\frac{r-3}{r-1}+o(1)} n^{2+o(1)}\right).$$

2.4.5 Proof of Theorem 2.1

In this section we restate and prove Theorem 2.1.

Theorem 2.1. [Algorithm] Let G be an r-rank n-vertex simple hypergraph of size p. Then, there exists a randomized algorithm that takes G as input and runs in time

$$\hat{O}_r\left(p + \min\left\{\lambda^{\frac{r-3}{r-1}}n^2, \frac{n^r}{\lambda^{\frac{r}{r-1}}}, \lambda^{\frac{5r-7}{4r-4}}n^{\frac{7}{4}}\right\}\right)$$

to return the connectivity λ of G with high probability. Moreover, the algorithm returns a min-cut in G with high probability.

Proof. We use the algorithm CXApproximation from Theorem 2.6 to obtain a value k such that $\lambda < k \leq 3\lambda$ in $O(p)$ time. If $k \leq 3r(4r^2)^r$, then the result follows by first obtaining a k-certificate of the hypergraph with the algorithm Certificate from Theorem 2.7 and then running Chekuri and Quanrud's algorithm CQMincut from Theorem 2.9 on the k-certificate. Henceforth, we assume that $k > 3r(4r^2)^r$ and consequently, $\lambda > r(4r^2)^r$.

Our algorithm finds an approximation for the min-cut capacity, then uses the k-certificate from Theorem 2.7 to reduce the size of the hypergraph to $O(\lambda n)$. At this point, we either run the min-cut algorithm from Theorem 2.5 or use the two algorithms we have described in sections 2.4.1 and 2.4.4, whichever is faster on the smaller hypergraph. We give a pseudocode of our approach in Algorithm 2.8. This algorithm uses the algorithms from Theorems 2.6, 2.7, 2.10 and 2.14.

```
MinCut(G):
    $k \leftarrow$ CXApproximation($G$)
    $G' \leftarrow$ Certificate($G, k$)
    $(C_1, V \setminus C_1) \leftarrow$ SmallSizeMinCut($G', r - \log(k/12r)/\log n$)
    $(C_2, V \setminus C_2) \leftarrow$ ExpDecompMinCut($G'$)
    $C \leftarrow \arg\min\{|\delta_G(C_1)|, |\delta_G(C_2)|\}$
    Return $(C, V \setminus C)$
```

Algorithm 2.8: MinCut

We first show that Algorithm 2.8 returns a min-cut. By Theorem 2.6, we have that $\lambda < k \leq 3\lambda$. Therefore, by Theorem 2.7, the subhypergraph G' is a simple hypergraph and every min-cut in G has capacity λ in G' and every other cut has capacity greater than λ in G'. Thus, the set of min-cuts in G' is exactly the same as the set of min-cuts in G. It remains to show that the cut $(C, V \setminus C)$ returned by the algorithm is a min-cut in G'. We distinguish two cases: (i) Suppose some min-cut in G' has size at most $r - \log(\lambda/4r)/\log n$.

We have that $r - \log(\lambda/4r)/\log n \leq r - \log(k/12r)/\log n$. Therefore, by Theorem 2.10, the call to SmallMinCut returns a min-cut in G', and thus $(C, V \setminus C)$ is a min-cut in G'. (ii) Suppose that every min-cut in G' has size greater than $r - \log(\lambda/4r)/\log n$. Then, by Theorem 2.14, the algorithm ExpDecompMinCut returns a min-cut in G' and thus, $(C, V \setminus C)$ is a min-cut in G'.

We now analyze the runtime of Algorithm 2.8. By Theorems 2.6 and 2.7, the computation of k and G' can be done in $O(p)$ time. Also by Theorem 2.7, the algorithm Certificate returns a hypergraph $G' = (V, E')$ with $\mathrm{vol}_{G'}(V) = \sum_{e \in E'} |e| = O(rkn)$. By Theorem 2.10, the runtime of SmallSizeMinCut$(G', r - \log(k/12r)/\log n)$ is

$$\tilde{O}_r\left(\left(r - \frac{\log\left(\frac{k}{12r}\right)}{\log n}\right)^{6r} kn\right) = \tilde{O}_r(\lambda n) = \tilde{O}_r(p). \tag{2.88}$$

By Theorems 2.6 and 2.7, the min-cut capacity in G' is still λ. Therefore, by Theorem 2.14, the runtime of ExpDecompMinCut(G') is

$$\tilde{O}_r\left(p + (\lambda n)^{1+o(1)} + \min\left\{\lambda^{\frac{r-3}{r-1}+o(1)} n^{2+o(1)}, \frac{n^{r+r\cdot o(1)}}{\lambda^{\frac{r}{r-1}}}, \lambda^{\frac{5r-7}{4r-4}+o(1)} n^{\frac{7}{4}+o(1)}\right\}\right) \tag{2.89}$$

$$= \hat{O}_r\left(p + \min\left\{\lambda^{\frac{r-3}{r-1}} n^2, \frac{n^r}{\lambda^{\frac{r}{r-1}}}, \lambda^{\frac{5r-7}{4r-4}} n^{\frac{7}{4}}\right\}\right). \tag{2.90}$$

Thus, the overall runtime of the algorithm is

$$O(p) + (r\lambda n)^{1+o(1)} + r^{O(r^3)} \min\left\{\lambda^{\frac{r-3}{r-1}+o(1)} n^{2+o(1)}, \left(\frac{n^r}{\lambda^{\frac{r}{r-1}}}\right) \log^{O(1)} n, \lambda^{\frac{5r-7}{4r-4}+o(1)} n^{\frac{7}{4}+o(1)} \log^{O(1)} n\right\} \tag{2.91}$$

$$= \hat{O}_r\left(p + \min\left\{\lambda^{\frac{r-3}{r-1}} n^2, \frac{n^r}{\lambda^{\frac{r}{r-1}}}, \lambda^{\frac{5r-7}{4r-4}} n^{\frac{7}{4}}\right\}\right). \tag{2.92}$$

QED.

Remark 2.4. We obtain the deterministic counterpart of Theorem 2.1 by using the deterministic version of ExpDecompMinCut in Algorithm 2.8 (that was discussed in Remark 2.3) and by using the deterministic algorithm for SmallMinCut described in Section 2.5 instead of the algorithm from section 2.4.1. The resulting algorithm has a run-time of $\hat{O}_r\left(p + \min\left\{\lambda n^2, \lambda^{\frac{r-3}{r-1}} n^2 + \frac{n^r}{\lambda}\right\}\right)$.

2.5 DETERMINISTIC ALGORITHM FOR FINDING SMALL-SIZED MIN-CUT

In this section, we consider the problem of finding a min-cut subject to an upper bound on the size of the smaller side of the cut. In particular, we prove the following algorithmic result, which can be viewed as a deterministic version of Theorem 2.10.

Lemma 2.9. Let $s \leq r$ be a positive integer, and let $G = (V, E)$ be an r-rank n-vertex hypergraph with m hyperedges that has a min-cut $(C, V \setminus C)$ with $|C| \leq s$. Then, there exists a deterministic algorithm which takes G and s as input and runs in time $O(2^{2s}n^s + 2^r m)$ to return a min-cut of G.

Proof. We begin by defining some functions that will be useful in our algorithm and analysis. For $S' \subseteq S \subseteq V$, let

$$g(S) := |\{e \in E \colon S \subseteq e\}|, \tag{2.93}$$

$$g'(S) := \begin{cases} 1 & \text{if } S \in E \\ 0 & \text{otherwise} \end{cases} \tag{2.94}$$

$$g_S(S') := |\{e \in E \colon e \cap S = S'\}|, \text{ and} \tag{2.95}$$

$$h_S(S') := |\{e \in E \colon e \cap S = S' \text{ and } e \setminus S \neq \emptyset\}|. \tag{2.96}$$

We first show that we can compute $g(S)$ and $g'(S)$ for all subsets $S \subseteq V$ of size at most s in time $O(n^s + 2^s m)$ using Algorithm 2.9.

```
MinCutHelper($G, s$):
  For all $S \subseteq V \colon |S| \leq s$
      $g[S] \leftarrow 0$
      $g'[S] \leftarrow 0$
  For all $e \in E$
      For all $S \subseteq e \colon |S| \leq s$
          $g[S] \leftarrow g[S] + 1$
      If $|e| \leq s$
          $g'[e] \leftarrow 1$
  Return $(g, g')$
```

Algorithm 2.9: MinCutHelper

We note that Algorithm 2.9 correctly computes $g(S)$ for each S since $g[S]$ will be incremented exactly once for each e containing S. It also correctly computes $g'(S)$, since $g'[S]$ will be set to 1 for an S of size at most s if and only if $e \in E$. Furthermore the algorithm spends $O(n^s)$ time initializing the arrays g and g' and $O(2^r m)$ iterating over subsets of the hyperedges (since a hyperedge of size at most r has at most 2^r subsets). Therefore, the algorithm runs in time $O(n^s + 2^r m)$.

Next, we solve the min s-sized cut problem using Algorithm 2.10.

```
SmallMinCut(G, s):
  $(g, g') \leftarrow$ MinCutHelper(G,s)
  For all $S \subseteq V \colon |S| \leq s$
      For all $S' \subset S$
          $g_S[S'] = \sum_{j=0}^{|S|-|S'|} (-1)^j \sum_{S'' \colon S' \subseteq S'' \subseteq S \text{ and } |S''|=|S'|+j} g[S'']$
          $h_S[S'] = g_S[S'] - g'[S']$
      $r[S] \leftarrow \sum_{S' \subseteq S} h_S[S']$
  $C \leftarrow \text{argmin}_{S \subseteq V \colon |S| \leq s} r[S]$
  Return $(C, V \setminus C)$
```

Algorithm 2.10: SmallMinCut

To see that Algorithm 2.10 is correct, we note that, by the principle of inclusion-exclusion, it correctly computes $g_S(S')$. To argue this, we show that a hyperedge e with $e \cap S = T \supset S'$ will not be counted in our summation for $g_S(S')$. Let $t = |T| - |S'|$. A set S'' such that $S' \subseteq S'' \subseteq S$ counts e in $g_S(S'')$ if and only if $S'' \subseteq T$. For every $j \in \{0, \ldots, t\}$, the number of sets S'' of size $|S'| + j$ such that $S' \subseteq S'' \subseteq T$ is $\binom{t}{j}$. Thus, the total contribution of e to our summation for $g_S[S']$ is

$$\sum_{j=0}^{t} (-1)^j \binom{t}{j} = 0. \tag{2.97}$$

A similar argument shows that the hyperedges which need to be counted in $g_S(S')$ are counted exactly once by the expression. Since $h_S(S')$ is $g_S(S')$ if $S' \notin e$ and $g_S(S') - 1$ otherwise, the algorithm correctly computes $h_S(S')$ as well. We note that every hyperedge $e \in \delta(S)$ must be counted in $h_S(S')$ for exactly one set $S' \subseteq S$ (namely $h_S(e \cap S)$). Therefore, $|\delta(S)| = \sum_{S' \subseteq S} h_S(S')$. Thus, we have that $r[S]$ correctly stores $\delta(S)$ for each $S \subseteq V$ of size at most s. Therefore, since the algorithm returns a set S with minimum $r[S]$, it returns a min s-sized cut.

The algorithm's outer **for** loop iterates over all subsets of V of size at most s. The number of such subsets is n^s. The inner for loop iterates over the subsets of a subset S of size at most s. The number of such subsets is at most 2^s. Thus, the inner **for** loop is executed $O(2^s n^s)$ times. Each iteration of the inner loop can be implemented to run in $O(2^s)$ time, since this is an upper bound on the number of sets S'' with $S' \subseteq S'' \subseteq S$, and thus on the number of terms in the double summation. Therefore, the total runtime of the outer loop is $O(2^{2s} n^s)$. Adding this to the runtime we computed for Algorithm 2.9 gives us an overall runtime of $O(2^{2s} n^s + 2^r m)$. QED.

2.6 TIGHT EXAMPLE FOR THE STRUCTURAL THEOREM

In this section, we show a tight example for Conclusion 2 in Theorem 2.2. Our example shows that there is no room to improve the factor $\lambda^{-1/(r-1)}$ in Conclusion 2 of Theorem 2.2. We prove the following lemma.

Lemma 2.10. There is a r-uniform simple hypergraph $G = (V, E)$ on n vertices and m hyperedges that satisfies the following conditions:

1. $\lambda(G) > n > r(4r^2)^r$,
2. the smaller side of every min-cut has size $\sqrt{n}$, and
3. the number of hyperedges in the union of all min-cuts is $\Theta(m/\lambda^{1/(r-1)})$.

Proof. Consider the complete r-uniform hypergraph $G_1 = (V_1, E_1)$ on $\sqrt{n}$ vertices. The degree of each vertex is $\Theta(n^{\frac{r-1}{2}})$. We make the following claim about G_1.

Claim 2.11. Every min-cut in G_1 has the smaller side to be a single vertex. Moreover, the capacity of the min-cut $\lambda(G_1) = \Theta(n^{\frac{r-1}{2}})$.

Proof. Consider a cut which has (say) t vertices on one side and $\sqrt{n}-t$ vertices on the other side. The capacity of this cut is

$$\sum_{a>0, b>0: a+b=r} \binom{t}{a}\binom{\sqrt{n}-t}{b}. \tag{2.98}$$

The above quantity is minimized when $t = 1$. Hence, the capacity of a min-cut is $\binom{\sqrt{n}-1}{r-1} = \Theta(n^{\frac{r-1}{2}})$. QED.

Now, we replace each vertex $u \in V_1$ with a set Q_u of $\sqrt{n}$ vertices. We do this in a way such that every vertex $v \in Q_u$ has degree $\Theta(n^{\frac{r-2}{2}})$ induced by the edges in E_1. This can be done as the total number of hyperedges incident on u is $\lambda(G_1)$. Because Q_u has $\sqrt{n}$ many vertices, we can replace u by a vertex $v \in Q_u$ in $\lambda(G_1)/\sqrt{n} = \Theta(n^{\frac{r-2}{2}})$ hyperedges. This ensures that each vertex $v \in Q_u$ has degree $\Theta(n^{\frac{r-2}{2}})$ induced by the hyperedges in E_1.

Having done so, we add all r-uniform hyperedges inside Q_u to make each Q_u a complete r-uniform hypergraph. We call this set of hyperedges as E_{Q_u}. This completes the description of $G = (V, E)$ on vertex set $V = \bigcup_{u \in V_1} Q_u$ of size n and $E = E_1 \cup \left(\bigcup_{u \in V_1} E_{Q_u}\right)$. Now we make the following claim.

Claim 2.12. Every min-cut in G is of the form $(Q_u, V \setminus Q_u)$ for some $u \in V_1$.

Proof. By a similar argument as in Claim 2.11, we see that the min-cut in the hypergraph $G_u = (Q_u, E_{Q_u})$ is obtained when a single vertex is cut from the set Q_u, and the capacity of the min-cut $\lambda(G_u) = \lambda(G_1) = \Theta(n^{\frac{r-1}{2}})$. By our construction of G, every cut of G that cuts G_u has capacity at least $\lambda(G_u) + \Theta(n^{\frac{r-2}{2}}) > \lambda(G_1)$. However, as proven in Claim 2.11, the capacity of a cut of the form $(Q_u, V \setminus Q_u)$ is $\lambda(G_1)$. QED.

Claim 2.12 implies that $\lambda := \lambda(G) = \lambda(G_1) = \Theta(n^{\frac{r-1}{2}})$. The number of hyperedges in G_1 is $m_1 = \binom{\sqrt{n}}{r}$. The number of hyperedges in G_u is $m_2' = \binom{\sqrt{n}}{r}$. Thus, the number of hyperedges in G is

$$m = m_1 + m_2'\sqrt{n} = \binom{\sqrt{n}}{r}(1+\sqrt{n}) = \Theta\left(n^{\frac{r+1}{2}}\right). \tag{2.99}$$

Claim 2.12 also implies that the number of hyperedges in the union of all min-cuts is exactly m_1. Now, we observe that

$$m_1 = \Theta\left(n^{\frac{r}{2}}\right) = \Theta\left(\frac{m}{\lambda^{\frac{1}{r-1}}}\right). \tag{2.100}$$

Moreover, in this graph G, we have that $\lambda > n > r(4r^2)^r$ by picking n to be large enough and $r \geq 3$.

QED.

2.7 MIN-CUT NON-TRIVIALITY FOR HYPERGRAPHS

In this section, we construct an example where $\lambda > |V|$ and even though every min-cut is non-trivial, the total number of hyperedges that take part in the union of all min-cuts is only a constant fraction of the total number of hyperedges.

If we consider a complete graph on n vertices, we see that every edge takes part in a min-cut. Hence the total number of edges that take part in the union of min-cuts is $m = \Theta(n^2)$. The issue with this example is that all min-cuts here are *trivial*, i.e., every min-cut has a single vertex as one side. We extend this example to hypergraphs such that (i) the capacity of the min-cut is $\Omega(n)$, (ii) all min-cuts are non-trivial, i.e., each min-cut has at least two vertices on each side, and (iii) the number of hyperedges taking part in the union of min-cuts is approximately m. We prove the following lemma.

Lemma 2.11. Let $n \geq 100$ be an even integer. Then, there is a simple hypergraph $G = (V, E)$ with $n + 3$ vertices and $m = \Theta(n^2)$ hyperedges such that

1. $\lambda(G) \geq n + 4$,

2. every min-cut has at least two vertices on both sides, and

3. the number of hyperedges in the union of all min-cuts is $\Theta(m)$.

Proof. We construct such a hypergraph G now. The vertex set V consists of $n+3$ vertices: $n/2$ vertices $\{u_1, \ldots, u_{n/2}\}$, $n/2$ vertices $\{v_1, \ldots, v_{n/2}\}$, and three special vertices $\{a, b, c\}$. We add the following 5-uniform hyperedges: For every pair of integers $1 \leq i < j \leq n/2$, we add the three hyperedges $\{u_i, v_i, u_j, v_j, a\}$, $\{u_i, v_i, u_j, v_j, b\}$ and $\{u_i, v_i, u_j, v_j, c\}$. We also add edges $\{u_i, v_i\}$ for each $i \in [n/2]$. The next three claims complete the proof of the lemma. QED.

Claim 2.13. The number of hyperedges in G is $\Theta(n^2)$.

Proof. For every two distinct pairs $\{u_i, v_i\}$ and $\{u_j, v_j\}$, there are three 5-uniform hyperedges. Hence the total number of 5-uniform hyperedges is $\Theta(n^2)$. The number of edges is $O(n)$. Hence the total number of hyperedges is $m = \Theta(n^2)$. QED.

Claim 2.14. The min-cut is of size $\lambda > n+3$. Moreover, each min-cut contains a pair $\{u_i, v_i\}$ on one side, and the rest of the vertices on the other side.

Proof. Let $(C, V \setminus C)$ be a min-cut in G. We first note that for every $i \in [n/2]$, the cut set $\delta(C)$ cannot contain the edge $\{u_i, v_i\}$. To see this, suppose $u_i \in C$, $v_i \in V \setminus C$, and assume without loss of generality that $|C| \leq |V \setminus C|$. We note that $V \setminus C \setminus \{v_i\}$ is non-empty since $|C| \leq |V \setminus C|$, and therefore $(C \cup \{v_i\}, V \setminus C \setminus \{v_i\})$ is a cut. Furthermore, this cut has strictly smaller capacity than $(C, V \setminus C)$, since it cuts no new hyperedges and it does not cut the edge $\{u_i, v_i\}$. Thus, we conclude that $(C, V \setminus C)$ does not cut any edges.

We note that one side of the min-cut $(C, V \setminus C)$ must include at most 1 vertex from among $\{a, b, c\}$. Without loss of generality assume that $|C \cap \{a, b, c\}| \leq 1$. We consider two cases. We will show that if $|C \cap \{a, b, c\} = 0$, then C must contain only a single pair $\{u_i, v_i\}$ and $|\delta(C)| > n+3$. Then we will show that the case where $|C \cap \{a, b, c\}| = 1$ cannot actually occur, because C is a min-cut.

Case 1: Suppose $|C \cap \{a, b, c\}| = 0$. Let k be the number of pairs $\{u_i, v_i\}$ with $u_i, v_i \in C$. Every hyperedge intersecting one of these pairs is cut by $(C, V \setminus C)$. The number of such hyperedges is minimized when $k = 1$. We note that for every pair $\{u_i, v_i\}$, there are $n/2 - 1$ pairs $\{u_j, v_j\}$ that share a 5-uniform hyperedge with $\{u_i, v_i\}$, and each pair shares three such hyperedges. Hence, $|\delta(C)| = |\delta(\{v_i, v_{i+1}\})| = 3(n/2 - 1) = 3n/2 - 3 > n+3$.

Case 2: Suppose $|C \cap \{a, b, c\}| = 1$. Without loss of generality, assume $a \in C$. Let k be the number of pairs $\{u_i, v_i\}$ with $u_i, v_i \in C$. We note that every hyperedge containing a pair $\{u_i, v_i\} \subseteq C$ and containing either vertex b or vertex c is in $\delta(C)$. There are $k(k-1)/2$ ways to pick two pairs inside C and $k(n/2-k)$ ways to pick a pair inside C and a pair outside C. Hence, $\delta(C)$ has $2(k(n/2-k)+k(k-1)/2)$ hyperedges which contain b or c. We also note that every hyperedge containing a pair $\{u_i, v_i\}$ with $u_i, v_i \in V \setminus C$ as well as the vertex a is in $\delta(C)$. Thus, $\delta(C)$ has $k(n/2-k)+(n/2-k)(n/2-k-1)/2$ hyperedges which contain a. Therefore, we have that $|\delta(C)| = 2(k(n/2-k)+k(k-1)/2)+k(n/2-k)+(n/2-k)(n/2-k-1)/2$ We note that for every $0 \leq k \leq n/2$, $\max\{k(k-1)/2, (n/2-k)(n/2-k-1)/2\} \geq (n/4)(n/4-1)/2 = (n^2-4n)/32 > 3n/2-3$. Thus, we have that $|\delta(C)| > 3n/2-3$. Since we showed in the previous paragraph that there is a cut in G of capacity $3n/2-3$, this contradicts the fact that $(C, V \setminus C)$ is a min-cut, and we conclude that this case cannot occur. QED.

Claim 2.15. The number of edges in the union of all min-cuts is $\Theta(n^2)$.

Proof. This follows from the proof of Claims 2.13 and 2.14. The hyperedges which take part in min-cuts are the 5-uniform hyperedges, and every such hyperedge takes part in some min-cut. Hence the claim follows. QED.

2.8 CONCLUSION AND OPEN PROBLEMS

Our runtime for hypergraph connectivity still falls short of the near-linear time that is known for graph min-cut. Thus, computing hypergraph connectivity in $\tilde{O}(p)$ time remains an open problem. Substantial progress on this problem has been made very recently. As we noted in Section 2.1.1, there is a randomized algorithm for computing hypergraph min-cut using a polylogarithmic number of max flow computations [7, 62, 64], and so, using the recent almost-linear time max flow algorithm of Chen, Kyng, Liu, Peng, Gutenberg, and Sachdeva [63], hypergraph min-cut (in arbitrary rank weighted hypergraphs) can be solved in time $O(p^{1+o(1)})$.

Another avenue for future research is designing efficient algorithms for Hypergraph-MinCut in alternative models of computation. In the rest of this section we consider Hypergraph-MinCut in the cut query and communication models, and discuss the difficulties in extending the approaches that work for Graph-MinCut in these models to hypergraphs.

Alternative computation models. The complexity of algorithms for submodular function minimization is measured not just in terms of their runtime but also in terms of the number of function evaluation queries they must perform. Therefore, when viewing Hypergraph-MinCut as a special case of submodular function minimization, it is natural to consider the *query complexity* of Hypergraph-MinCut. In the cut query model, the input is a hypergraph G given by an oracle for the cut function d_G of the hypergraph. The cut query complexity of an algorithm in this model is simply the number of calls that the algorithm makes to the oracle for d_G, regardless of how much additional computation it performs.

Another computation model that is closely related to the cut query model is the communication model. In the communication model, there are two agents, Alice and Bob. Alice is given a hypergraph $G_1 = (V, E_1)$ and Bob is given a hypergraph $G_2 = (V, E_2)$, where $E_1 \cap E_2 = \emptyset$. Their goal is to solve some problem $\mathcal{P}$ (e.g. min-cut) in the hypergraph $G := (V, E_1 \cup E_2)$ using as few bits of communication between the two of them as possible (regardless of the amount of additional computation that this requires each of them to perform). The deterministic communication complexity of $\mathcal{P}$ is the number of bits of information that Alice and Bob must exchange to solve $\mathcal{P}$. The randomized communication complexity of $\mathcal{P}$ is the number of bits of information that Alice and Bob must exchange to compute an answer to $\mathcal{P}$ which is correct with probability at least $2/3$. The following lemma relating the two models can be proved via a simple simulation argument.

Lemma 2.12 (Reduction from communication to cut query model)**.** For every function $f(n, r)$, if Hypergraph-MinCut can be computed in $O(f(n, r))$ cut queries in an n-vertex r-rank hypergraph, then Hypergraph-MinCut can be computed in $\tilde{O}(f(n, r))$ bits of communication.

In graphs, the query complexity and communication complexity of min-cut are both $\tilde{O}(n)$ [95], and this result is tight up to logarithmic factors [54, 111]. For hypergraphs, the best upper bound known on the query complexity as well as the communication complexity comes from work on submodular function minimization. There is an algorithm for submodular function minimization which makes $O(n^2 \log n)$ queries to the evaluation oracle [52], which implies that the query complexity of Hypergraph-MinCut is $\tilde{O}(n^2)$. Therefore, the following question is of interest (Note that, by Lemma 2.12, an algorithm using $o(n^2)$ cut queries would immediately imply an algorithm using $o(n^2)$ communication. The reverse is not true.):

Question 2.1. Can Hypergraph-MinCut be solved in $o(n^2)$ cut queries or bits of communication?

In fact, for arbitrary rank weighted hypergraphs, it is not even known how to obtain a constant factor approximation for Hypergraph-MinCut in $o(n^2)$ bits of communication. Thus, the following weaker question is also of interest.

Question 2.2. Can an $O(1)$ approximation for Hypergraph-MinCut be computed in $o(n^2)$ cut queries or bits of communication?

Known result for graphs and barriers for hypergraphs. Rubinstein, Schramm, and Weinberg gave a randomized algorithm which finds a min-cut in a simple graph in $\tilde{O}(n)$ cut-queries [95]. Their result shows that the query complexity of graph min-cut is $\tilde{O}(n)$, and therefore so is the communication complexity. The proof from [95] does not seem to generalize to hypergraphs for two reasons: Firstly, their proof relies on the following structural result which does not hold for hypergraphs:

Lemma 2.13. [95] Let $G = (V, E)$ be an unweighted graph with minimum degree d and min-cut value c. Let $\epsilon < 1$ be a constant. Let $\mathcal{C}$ be the set of all non-singleton approximate-minimum cuts in the graph, with cut value at most $c + \epsilon d$. Then the total number of edges that participate in cuts in $\mathcal{C}$ is $O(n)$.

We note that this lemma fails to hold even for exact min-cuts in rank 3 hypergraphs. Consider a rank 3 hypergraph formed by taking two complete 3-uniform hypergraphs X and Y, each on $n/2$ vertices, and then adding $n^{1.5}$ size 3 hyperedges each of which intersects both X and Y (this is possible, since the number of potential rank 3 hyperedges intersecting X and Y is $n\binom{n/2}{2}$). We note that (for sufficiently large n) (X, Y) is the only min-cut in this hypergraph, since every cut which splits X or Y must cut at least $\binom{n/2-1}{2} = \Omega(n^2)$ hyperedges. But the cut (X, Y) cuts $\Omega(n^{1.5})$ hyperedges by definition, so it is not true that the number of hyperedges that participate in min-cuts is $O(n)$. We note that the cut (X, Y) in this example is perfectly balanced, meaning that even if we exclude cuts where one side of the cut has a constant number of vertices (rather than just excluding singleton cuts), the lemma would still not hold.

The second reason that [95]'s approach fails is that their algorithm involves learning all of the edges in the graph (after subsampling and contracting) which is not possible with hypergraph cut queries. In particular, we show the following theorem:

Theorem 2.15. For every $n \geq 50$, there exist two distinct n-vertex hypergraphs G and H with the same vertex set V such that $d_G(\delta(S)) = d_H(\delta(S))$ for every $S \subseteq V$.

This theorem follows via an information theoretic argument from the following lemma, which says that all cut queries in 3-uniform hypergraphs can be expressed in terms of singleton and pair queries:

Lemma 2.14. Let H be a 3-uniform hypergraph. Then, knowing the cut function values of all singletons and pairs of vertices of H allows one to compute the cut function value on all subsets of vertices with no additional queries to the cut function.

Proof. Let $H = (V, E)$ be a 3-uniform hypergraph, let $S \subseteq V$ be arbitrary such that $|S| \geq 3$, and let $k := |S|$. We will show that $d(S) = (\sum_{u,v \in S} d(\{u, v\})) - (k-2) \sum_{v \in S} d(\{v\})$

We consider the hyperedges of H based on how many vertices of S they intersect.

- Hyperedges that contain no vertices from S make no contribution to $(\sum_{u,v \in S} d(\{u, v\})) - (k-2) \sum_{v \in S} d(\{v\})$, and these edges are not in $\delta(S)$.
- Hyperedges that contain exactly one vertex from S, call it x, are counted $k-1$ times in $\sum_{\{u,v\} \subseteq S} d(\{u, v\})$ (because there are $k-1$ ways to choose a vertex to pair x with). These hyperedges are counted once in $\sum_{v \in S} d(\{v\})$ (because the only single vertex cut from S they cross is the one containing just x), and thus they are also counted exactly once in $(\sum_{u,v \in S} d(\{u, v\})) - (k-2) \sum_{v \in S} d(\{v\})$, and this is appropriate, because each of these edges is in $\delta(S)$.
- Hyperedges that contain exactly two vertices from S, call them x and y, are counted $2k-3$ times in $\sum_{\{u,v\} \subseteq S} d(\{u, v\})$, because there $k-2$ pairs of the form $\{x, z\}$ such that $z \in S \setminus \{x, y\}$, $k-2$ pairs of the form $\{y, z\}$ such that $z \in S \setminus \{x, y\}$, and one pair $\{x, y\}$, and these $2k-3$ pairs are all the ones which involve x, y, or both (and since H is 3-uniform, $d(\{u, v\})$ will count every hyperedge containing u, v or both). These hyperedges are counted twice in $\sum_{v \in S} d(\{v\})$, once in $d(\{x\})$ and once in $d(\{y\})$. Thus, each of these hyperedges is counted exactly once in $(\sum_{u,v \in S} d(\{u, v\})) - (k-2) \sum_{v \in S} d(\{v\})$, which again is correct, since each of these edges is in $\delta(S)$.
- Hyperedges with all three vertices, call them x, y, and z, in S are counted $3k-6$ times in $\sum_{\{u,v\} \subseteq S} d(\{u, v\})$, because there are $3(k-3)$ ways to pick one of the three vertices and pair it with a vertex not in this hyperedge, plus 3 ways to pair the vertices in the hyperedge with each other. On the other hand, each of these edges is counted 3 times in $\sum_{v \in S} d(\{v\})$, once for each of their vertices, and thus such edges do not contribute to $(\sum_{u,v \in S} d(\{u, v\})) - (k-2) \sum_{v \in S} c(\{v\})$ at all. This matches the fact that edges of this type are not in $\delta(S)$, since they are fully contained in S.

Thus, we have concluded that each hyperedge in $\delta(S)$ is counted exactly once in the formula $(\sum_{u,v \in S} d(\{u, v\})) - (k-2) \sum_{v \in S} c(\{v\})$, and hyperedges not in $\delta(S)$ are not counted at all in this formula, meaning that this formula correctly gives $d(S)$, as desired. QED.

We now prove Theorem 2.15 using Lemma 2.14.

Proof of Theorem 2.15. Let $n \geq 50$ be a positive integer, and let $\mathcal{G}$ be the set of all simple 3-uniform hypergraphs on vertex set $[n]$. Let $\mathcal{D} := \{d_G \colon G \in \mathcal{G}\}$ be the set of all cut functions corresponding to hypergraphs in $\mathcal{G}$. By Lemma 2.14, each function in $\mathcal{D}$ is uniquely determined by its outputs on subsets of $[n]$ of size 1 or 2. The number of such subsets is $n+\binom{n}{2} = \frac{n(n+1)}{2} < n^2$. A 3-uniform hypergraph can have at most $n-2+2\binom{n-2}{2} = (n-2)^2 < n^2$ hyperedges intersecting a given subset of size at most 2. Thus, since the hypergraphs in $\mathcal{G}$ are simple, their cut functions can take on at most n^2 distinct values for a given subset of $[n]$ of size at most 2, and there are fewer than $(n^2)^{n^2}$ ways to choose a value for the cut function for every subset of $[n]$ of size 1 or 2. Thus, $|\mathcal{D}| \leq n^{2n^2} = 2^{2n^2 \log n}$. The number of 3-uniform hypergraphs on vertex set $[n]$ is exactly $2^{\binom{n}{3}}$. Since $n \geq 50$, we have that $\binom{n}{3} > 2n^2 \log n$, and thus $|\mathcal{G}| = 2^{\binom{n}{3}} > 2^{2n^2 \log n} > |\mathcal{D}|$. Since there are more simple 3-uniform hypergraphs on n vertices than cut functions for these hypergraphs, we conclude that two of the hypergraphs in $\mathcal{G}$ must share the same cut function. QED.

CHAPTER 3: ENUMERATING MIN-CUT-SETS AND MIN-k-CUT-SETS IN HYPERGRAPHS

In this chapter we consider the problem of deterministically enumerating all minimum k-cut-sets in a given hypergraph for a fixed k. The input here is a hypergraph $G = (V, E)$ with non-negative hyperedge costs. A subset $F \subseteq E$ of hyperedges is a k-cut-set if the number of connected components in $G - F$ is at least k and it is a minimum k-cut-set if it has the least cost among all k-cut-sets. For fixed k, we call the problem of finding a minimum k-cut-set Hypergraph-k-Cut and the problem of enumerating all minimum k-cut-sets Enum-Hypergraph-k-Cut. The special cases of Hypergraph-k-Cut and Enum-Hypergraph-k-Cut restricted to graph inputs are well-known to be solvable in (randomized as well as deterministic) polynomial time [65, 68, 72, 106]. In contrast, it is only recently that polynomial-time algorithms for Hypergraph-k-Cut were developed [35, 36, 73]. The randomized polynomial-time algorithm for Hypergraph-k-Cut that was designed in 2018 [35] showed that the number of minimum k-cut-sets in a hypergraph is $O(n^{2k-2})$, where n is the number of vertices in the input hypergraph, and that they can all be enumerated in randomized polynomial time, thus resolving Enum-Hypergraph-k-Cut in randomized polynomial time. A deterministic polynomial-time algorithm for Hypergraph-k-Cut was subsequently designed in 2020 [73], but it is not guaranteed to enumerate all minimum k-cut-sets. In this chapter, we give the first deterministic polynomial-time algorithm to solve Enum-Hypergraph-k-Cut (this is non-trivial even for $k = 2$). Our algorithm is based on new structural results that allow for efficient recovery of all minimum k-cut-sets by solving minimum (S, T)-terminal cuts. Our techniques give new structural insights even for enumerating all minimum cut-sets (i.e., minimum 2-cut-sets) in a given hypergraph. The results in this chapter are based on joint work with Chandrasekaran and Wang and appeared in SODA '22 [74]. The research in this chapter was supported in part by NSF grants CCF-1814613 and CCF-1907937.

3.1 INTRODUCTION

A hypergraph $G = (V, E)$ consists of a finite set V of vertices and a finite set E of hyperedges where each hyperedge $e \in E$ is a subset of V. We consider the problem of enumerating all optimum solutions to the Hypergraph-k-Cut problem when k is a fixed constant. In Hypergraph-k-Cut, the input consists of a hypergraph $G = (V, E)$ with non-negative hyperedge-costs $c : E \to \mathbb{R}_+$ and a positive integer k. The objective is to find a minimum-cost subset of hyperedges whose removal results in at least k connected components. We will call a subset of hyperedges whose removal results in at least k connected components a *k-cut-set* and we will call a minimum-cost k-cut-set a *minimum k-cut-set*; for $k = 2$, we will refer to a 2-cut-set as simply a *cut-set* and a minimum-cost cut-set as a *minimum cut-set*. The central problem of interest to this chapter is that of enumerating all minimum k-cut-sets in a given hypergraph with non-negative hyperedge-costs—we will denote this problem as ENUM-HYPERGRAPH-k-CUT. Throughout, we will consider k to be a fixed constant integer (e.g., $k = 2, 3, 4, ...$). We will denote HYPERGRAPH-k-CUT and ENUM-HYPERGRAPH-k-CUT for graph inputs as GRAPH-k-CUT and ENUM-GRAPH-k-CUT respectively. We note that the case of $k = 2$ corresponds to global minimum cut which will be discussed shortly.

Partitioning formulation. Even for $k = 2$, there is a fundamental structural difference between HYPERGRAPH-k-CUT and GRAPH-k-CUT, which is especially evident when attempting to enumerate all optimum solutions. In order to illustrate this difference, we discuss an equivalent partitioning formulation of HYPERGRAPH-k-CUT. In this equivalent formulation, the objective is to find a partition of the vertex set V into k non-empty sets $V_1, V_2, \dots, V_k$ so as to minimize the cost of hyperedges that cross the partition. A hyperedge $e \in E$ is said to *cross a partition* $V_1, V_2, \dots, V_k$ if it has vertices in at least two parts, that is, there exist distinct $i, j \in [k]$ such that $e \cap V_i \neq \emptyset$ and $e \cap V_j \neq \emptyset$. We will denote a partition of V into k non-empty parts as a *k-partition* and a 2-partition as a *cut*. The cost of a k-partition is the sum of the cost of hyperedges crossing the partition. A k-partition with minimum cost is said to be a *minimum k-partition*. We will denote the cost of a 2-partition as its cut value and a minimum 2-partition as a minimum cut.

By definition, the number of minimum k-cut-sets is at most the number of minimum k-partitions. Moreover, for a connected *graph*, the number of minimum k-partitions is $O(n^k)$ for constant k, where n is the number of vertices (i.e., the number of minimum k-partitions is polynomial since k is a constant) [65, 66, 67]. However, for a connected[4] hypergraph, the number of minimum k-partitions could be exponential while the number of minimum k-cut-sets is only polynomial. For example, consider the *spanning-hyperedge-example*: this is the n-vertex hypergraph $G = (V, E)$ that consists of only one hyperedge e where $e = V$ with the cost of the hyperedge e being one. This hypergraph is connected and has only one minimum k-cut-set but $\Theta(k^n)$ minimum k-partitions (i.e., an exponential number of minimum k-partitions even for $k = 2$). Thus, if we are hoping for polynomial-time algorithms to enumerate all optimum solutions to HYPERGRAPH-k-CUT, then we cannot aim to enumerate all minimum k-partitions (in contrast to connected graphs). This is the

[4]A hypergraph is defined to be *connected* if the every cut has at least one hyperedge crossing it.

reason for defining Enum-Hypergraph-k-Cut as the problem of enumerating all minimum k-cut-sets as opposed to enumerating all minimum k-partitions. For connected graphs, the two definitions are indeed equivalent.

Graph-k-Cut for $k = 2$ is the minimum cut problem in graphs which is well-known to be solvable in polynomial time. Although the minimum cut problem in graphs has been extensively studied, enumerating all minimum cut-sets in a graph in deterministic polynomial time is already non-trivial. Dinitz, Karzanov, and Lomonosov [15] constructed a compact representation for all minimum cuts in a connected graph (known as the *cactus representation*) which showed that the number of minimum cuts in a connected graph is at most $\binom{n}{2}$ and that they can all be enumerated in deterministic polynomial time. For constant $k \geq 3$, the number of minimum k-partitions in a connected graph is $O(n^k)$—this bound is tight and is a consequence of a recent improved analysis of a random contraction algorithm to solve Graph-k-Cut [65, 66, 67]; the same random contraction algorithm can also be used to enumerate all minimum k-partitions in connected graphs in randomized polynomial time. Deterministic polynomial-time algorithms to enumerate all minimum k-partitions in connected graphs are also known. We discuss other techniques—both randomized and deterministic—for enumerating minimum cuts and minimum k-partitions in graphs in Section 3.1.2.

Hypergraph-k-Cut is a natural generalization of Graph-k-Cut. Hypergraph-k-Cut for $k = 2$ is the minimum cut problem in hypergraphs which is well-known to be solvable in polynomial time [112]. Once again, enumerating all minimum cut-sets in a hypergraph in deterministic polynomial-time is already non-trivial. We encourage the reader to pause and think briefly about possible approaches to enumerate all minimum cut-sets in a hypergraph before reading further. There exists a compact representation of all minimum cut-sets in a hypergraph [38]—namely the *hypercactus representation*—which can be used to show that the number of minimum cut-sets in a hypergraph is at most $\binom{n}{2}$ and that they can all be enumerated in deterministic polynomial time. To the best of the authors' knowledge, this is the only known technique for efficient deterministic enumeration of all minimum cut-sets in a hypergraph.

Hypergraph-k-Cut is a special case of Submodular-k-Partition (e.g., see [27, 73, 113, 114]). Owing to this connection, the complexity of Hypergraph-k-Cut for every fixed $k \geq 3$ has been an intriguing open question until recently. A randomized polynomial-time algorithm for Hypergraph-k-Cut was designed in 2018 by Chandrasekaran, Xu, and Yu [35]. The analysis of this algorithm showed that the number of minimum k-cut-sets is $O(n^{2k-2})$, where n is the number of vertices in the input hypergraph (i.e., the number of minimum k-cut-sets is polynomial), and that they can all be enumerated in randomized polynomial time (also see [36]). Subsequently, Chandrasekaran and Chekuri designed a deterministic polynomial-time algorithm for HYPERGRAPH-k-CUT in 2020 [73]. However, their deterministic algorithm is guaranteed to identify only one minimum k-cut-set and not all. The next natural question is whether all minimum k-cut-sets can be enumerated in deterministic polynomial time—namely, can we solve ENUM-HYPERGRAPH-k-CUT in deterministic polynomial time?

As mentioned earlier, the only known technique for ENUM-HYPERGRAPH-k-CUT for $k = 2$ is via the hypercactus representation which does not seem to generalize to $k \geq 3$ (in fact, it is unclear if cactus representation generalizes to $k \geq 3$ even in graphs). Moreover, all deterministic techniques for ENUM-GRAPH-k-CUT address the problem of enumerating all minimum k-partitions in connected graphs—see Section 3.1.2; hence, all these techniques fail for ENUM-HYPERGRAPH-k-CUT (as seen from the spanning-hyperedge-example). For hypergraphs, we necessarily have to work with minimum k-cut-sets as opposed to minimum k-partitions. Working with minimum k-cut-sets as opposed to minimum k-partitions in the deterministic setting is a technical challenge that has none of the previous works have undertaken (even for graphs). We overcome this technical challenge in this work. We adapt Chandrasekaran and Chekuri's deterministic approach for HYPERGRAPH-k-CUT and augment it with structural results for minimum k-cut-sets to prove our main result stated below.

Theorem 3.1. For every fixed k, there is a deterministic polynomial-time algorithm for ENUM-HYPERGRAPH-k-CUT.

Although we chose to highlight the above algorithmic result in this introduction, we emphasize that the structural theorems that form the backbone of the algorithmic result are our main technical contributions (see Theorems 3.2 and 3.3). We discuss these structural theorems in the technical overview section. By tightening the proof technique of one of our structural theorems for $k = 2$, we obtain an arguably elegant structural explanation for the number of minimum cut-sets in a hypergraph being at most $\binom{n}{2}$—see Theorem 3.4. Theorem 3.4 leads to an alternative deterministic polynomial-time algorithm to enumerate all minimum cut-sets in a hypergraph (that is relatively simpler than computing a hypercactus representation). We believe that our structural theorems are likely to be of independent interest in the theory of hypergraphs.

3.1.1 Technical Overview and Main Structural Results

We focus on the unit-cost variant of Enum-Hypergraph-k-Cut in the rest of this chapter for the sake of notational simplicity. Throughout, we will allow multigraphs and hence, this is without loss of generality. Our algorithms extend in a straightforward manner to arbitrary hyperedge costs. They rely only on minimum (s,t)-terminal cut computations and hence, they are strongly polynomial-time algorithms.

A key algorithmic tool will be the use of terminal cuts. We need some notation. Let $G = (V, E)$ be a hypergraph. Throughout this chapter, n will denote the number of vertices in G and $p := \sum_{e\in E} |e|$ will denote the representation size of G. We will denote a partition of the vertex set into h non-empty parts by an ordered tuple $(V_1, \ldots, V_h)$. For a non-empty proper subset U of vertices, we will use $\overline{U}$ to denote $V \setminus U$, $\delta(U)$ to denote the set of hyperedges crossing the 2-partition $(U, \overline{U})$, and $d(U) := |\delta(U)|$. We recall that $\delta(U) = \delta(\overline{U})$, so we will use $d(U)$ to denote the cost of the cut $(U, \overline{U})$. More generally, given a partition $\mathcal{P} = (V_1, V_2, \ldots, V_h)$, we denote the set of hyperedges crossing the partition by $\delta(V_1, V_2, \ldots, V_h)$ (also by $\delta(\mathcal{P})$ for brevity) and the number of hyperedges crossing the partition by $\text{cost}(V_1, V_2, \ldots, V_h) := |\delta(V_1, V_2, \ldots, V_h)|$ (also by $\text{cost}(\mathcal{P})$ for brevity). Let S, T be disjoint non-empty subsets of vertices. A 2-partition $(U, \overline{U})$ is an (S,T)-terminal cut if $S \subseteq U \subseteq V \setminus T$. Here, the set U is known as the source set and the set $\overline{U}$ is known as the sink set. A minimum-cost (S,T)-terminal cut is known as a *minimum* (S,T)*-terminal cut.* Since there could be multiple minimum (S,T)-terminal cuts, we will be interested in *source minimal* minimum (S,T)-terminal cuts and *source maximal* minimum (S,T)-terminal cuts: a minimum (S,T)-terminal cut $(U, \overline{U})$ is a source minimal minimum (S,T)-terminal cut if there does not exist a minimum (S,T)-terminal cut $(U', \overline{U'})$ such that $U' \subsetneq U$ and is a source maximal minimum (S,T)-terminal cut if there does not exist a minimum (S,T)-terminal cut $(U', \overline{U'})$ with $U \subsetneq U'$. For every pair of disjoint non-empty subsets S and T of vertices, there exists a unique source minimal minimum (S,T)-terminal cut and it can be found in deterministic polynomial time via standard maxflow algorithms; a similar result holds for source maximal minimum (S,T)-terminal cuts.

Our algorithm is inspired by the divide and conquer approach introduced by Goldschmidt and Hochbaum for Graph-k-Cut [72]. This approach was generalized by Kamidoi, Yoshida, and Nagamochi to solve Enum-Graph-k-Cut [68] and by Chandrasekaran and Chekuri to solve Hypergraph-k-Cut [73], both in deterministic polynomial time. The techniques of [72] and [68] are not applicable to Enum-Hypergraph-k-Cut since they are tailored to graphs and do not extend to hypergraphs. We describe the details of the divide and conquer approach for Hypergraph-k-Cut due to Chandrasekaran and Chekuri [73]. The goal here is to identify one part of some fixed minimum k-partiton $(V_1, V_2, \ldots, V_k)$, say V_1 without loss of generality, and then recursively find a minimum $(k-1)$-partition in the subhypergraph $G[\overline{V_1}]$, where $G[\overline{V_1}]$ is the hypergraph obtained from G by discarding the vertices in V_1 and by discarding all hyperedges that intersect V_1. Now, how does one find such a part V_1? Chandrasekaran and Chekuri proved a key structural theorem for this: Suppose $(V_1, \ldots, V_k)$ is a V_1-*maximal* minimum k-partition—i.e., there is no other minimum k-partition $(V_1', \ldots, V_k')$ such that V_1 is a proper subset of V_1'. Then, they showed that for every subset $T \subseteq \overline{V_1}$ such that $T \cap V_j \neq \emptyset$ for all $j \in \{2, \ldots, k\}$, there exists a subset $S \subseteq V_1$ of size at most $2k-2$ such that $(V_1, \overline{V_1})$ is the source maximal minimum (S,T)-terminal cut. A consequence of this structural theorem is that if we compute the collection $\mathcal{C}$ consisting of the source side of the source maximal minimum (S,T)-terminal cut for all possible pairs (S,T) of disjoint subsets of vertices S and T with $|S| \leq 2k-2$ and $|T| \leq k-1$, then the set V_1 will be in this collection $\mathcal{C}$ (by applying the structural theorem to a set T of size $k-1$ with $|T \cap V_j| = 1$ for all $j \in \{2, \ldots, k\}$). Moreover, the size of the collection $\mathcal{C}$ is only $O(n^{3k-3})$. Hence, recursing on $G[\overline{U}]$ for each set U in the collection $\mathcal{C}$ will identify a minimum k-partition within a total run-time of $n^{O(k^2)}$ source maximal minimum (S,T)-terminal cut computations.

The limitation of the structural theorem of Chandrasekaran and Chekuri [73] is that it aims to recover a minimum k-partition and in particular, a V_1-maximal minimum k-partition. For the purposes of enumerating all minimum k-cut-sets, this is insufficient as we have seen from the spanning-hyperedge-example. In particular, their structural theorem cannot be used to even enumerate all minimum cut-sets in a hypergraph. We prove two structural theorems that will help in enumerating minimum k-cut-sets. We describe these structural theorems now.

Our goal is to deterministically enumerate a polynomial-sized family $\mathcal{F}$ of k-cut-sets such that $\mathcal{F}$ contains all minimum k-cut-sets. Let F be an arbitrary minimum k-cut-set. Since F is a minimum k-cut-set, there exists a minimum k-partition $(V_1, \ldots, V_k)$ such that $F = \delta(V_1, \ldots, V_k)$. We note that $d(V_1) \leq |F|$ by definition of the hypergraph cut function $d : 2^V \to \mathbb{R}$. We distinguish two cases:

Case 1. Suppose $d(V_1) < |F|$. In order to identify minimum k-cut-sets F that have this property, we show the following structural theorem.

Theorem 3.2. Let $G = (V, E)$ be a hypergraph and let OPT_k be the value of a minimum k-cut-set in G for some integer $k \geq 2$. Suppose $(U, \overline{U})$ is a 2-partition of V with $d(U) < OPT_k$. Then, for every pair of vertices $s \in U$ and $t \in \overline{U}$, there exist subsets $S \subseteq U \setminus \{s\}$ and $T \subseteq \overline{U} \setminus \{t\}$ with $|S| \leq 2k-3$ and $|T| \leq 2k-3$ such that $(U, \overline{U})$ is the unique minimum

$(S \cup \{s\}, T \cup \{t\})$-terminal cut in G.

The advantage of this structural theorem is that it allows for a recursive approach to enumerate a polynomial-sized family of minimum k-cut-sets containing F under the assumption that $d(V_1) < |F| = OPT_k$ (similar to the approach of Chandrasekaran and Chekuri).

The drawback of this structural theorem is that it only addresses the case of $d(V_1) < |F|$. It is possible that the minimum k-cut-set F satisfies $d(V_1) = |F|$. For example, consider the problem of enumerating all minimum cut-sets in a hypergraph (i.e., Enum-Hypergraph-k-Cut for $k = 2$)—Theorem 3.2 does not help in this case since there will be no cut $(U, \overline{U})$ with $d(U) < \mathrm{OPT}_2$. This motivates the second case.

Case 2. Suppose $d(V_1) = |F|$. In this case, we need to enumerate a polynomial-sized family of k-cut-sets containing F, but we cannot hope to enumerate all minimum k-partitions $(V_1', \ldots, V_k')$ for which $F = \delta(V_1', \ldots, V_k')$ (e.g., again consider the spanning-hyperedge-example for $k = 2$ for which the unique minimum cut-set F has $|F| = d(V_1)$ for exponentially many minimum cuts (V_1, V_2) and hence, we cannot hope to enumerate all minimum cuts). We observe that if $d(V_1) = |F|$, then the set F of hyperedges should be equal to the set of hyperedges crossing $(V_1, \overline{V_1})$, i.e., $\delta(V_1) = F = \delta(V_1, \ldots, V_k)$. We show the following structural theorem to exploit this observation.

Theorem 3.3. Let $G = (V, E)$ be a hypergraph, $k \geq 2$ be an integer, and $\mathcal{P} = (V_1, \ldots, V_k)$ be a minimum k-partition such that $\delta(V_1) = \delta(\mathcal{P})$. Then, for all subsets $T \subseteq \overline{V_1}$ such that $T \cap V_j \neq \emptyset$ for all $j \in \{2, 3, \ldots, k\}$, there exists a subset $S \subseteq V_1$ with $|S| \leq 2k - 1$ such that the source minimal minimum (S, T)-terminal cut $(A, \overline{A})$ satisfies $\delta(A) = \delta(V_1)$ and $A \subseteq V_1$.

We recall that for fixed disjoint subsets $S, T \subseteq V$, the source minimal minimum (S, T)-terminal cut is unique. We emphasize the main feature of Theorem 3.3: it aims to recover only the hyperedges crossing the cut $(V_1, \overline{V_1})$ but not the cut $(V_1, \overline{V_1})$ itself. It shows the existence of a small-sized witness which allows us to recover $\delta(V_1)$—namely a pair (S, T) with $|S|, |T| = O(k)$ for which $\delta(V_1)$ is the cut-set of the source minimal minimum (S, T)-terminal cut. In this sense, Theorem 3.3 addresses the drawback of Theorem 3.2.

Theorems 3.2 and 3.3 can be used to design a recursive algorithm that enumerates all minimum k-cut-sets in deterministic polynomial time (along the lines of the algorithm of Chandrasekaran and Chekuri described above). Here, we describe a more straightforward non-recursive deterministic polynomial-time algorithm. For each pair of subsets of vertices S, T of size at most $2k-1$, we compute the source minimal minimum (S, T)-terminal cut $V_{S,T}$; if $G - \delta(V_{S,T})$ has at least k connected components, then we add $\delta(V_{S,T})$ to the candidate family $\mathcal{F}$; otherwise, we add $V_{S,T}$ to the collection $\mathcal{C}$. Next, we consider all possible k-partitions $(U_1, \ldots, U_k)$ of the vertex set where all sets $U_1, \ldots, U_k$ are in the collection $\mathcal{C}$ and add the set $\delta(U_1, \ldots, U_k)$ of hyperedges to the family $\mathcal{F}$. We now sketch the argument to show that the family $\mathcal{F}$ contains the (arbitrary) minimum k-cut-set F. Recall that there exists a minimum k-partition $(V_1, \ldots, V_k)$ such that F is the set of hyperedges crossing this k-partition, i.e., $F = \delta(V_1, \ldots, V_k)$. We have two possibilities: (1) if $d(V_i) < |F|$ for every $i \in [k]$, then by Theorem 3.2, every set V_i is in the collection $\mathcal{C}$ (by applying Theorem 3.2 to $(U = V_i, \overline{U} = \overline{V_i})$ and arbitrary vertices $s \in V_i, t \in \overline{V_i}$), and hence $F \in \mathcal{F}$; (2) if $d(V_i) = |F|$ for some $i \in [k]$, then by Theorem 3.3, one of the sets $V_{S,T}$ has $\delta(V_{S,T}) = \delta(V_i) = F$ and hence, once again $F \in \mathcal{F}$. We can prune the family $\mathcal{F}$ to return the subfamily of minimum k-cut-sets in it. The size of the collection $\mathcal{C}$ is $O(n^{4k-2})$ and the size of the family $\mathcal{F}$ is $O(n^{4k^2})$. The run-time is $O(n^{4k-2})T(n,p) + O(n^{4k^2})$, where $T(n,p)$ is the time complexity for computing the source minimal minimum (s,t)-terminal cut in a n-vertex hypergraph of size p.

Additional consequence of Theorem 3.3. Theorem 3.3 is the technical novelty of this work. We emphasize another structural consequence of Theorem 3.3 by using it to bound the number of minimum cut-sets in a hypergraph. Let t be an arbitrary vertex in the hypergraph $G = (V, E)$. Consider the sets

$$\mathcal{H} := \{U \subseteq V \setminus \{t\} : (U, \overline{U}) \text{ is a minimum cut in } G\} \text{ and} \tag{3.1}$$

$$\mathcal{M} := \{\delta(U) : U \in \mathcal{H}\}. \tag{3.2}$$

We note that $\mathcal{M}$ is the family of all minimum cut-sets in the hypergraph. By applying Theorem 3.3 for $k = 2$ and $T = \{t\}$, we obtain that for every set $U \in \mathcal{H}$, there exists a subset $S \subseteq U$ with $|S| \leq 3$ such that the source minimal minimum $(S, \{t\})$-terminal cut $(A, \overline{A})$ satisfies $\delta(A) = \delta(U)$. Consequently, the size of the set $\mathcal{M}$ is at most the number of possible ways to choose a non-empty subset $S \subseteq V \setminus \{t\}$ of size at most 3 which is $\binom{n-1}{1} + \binom{n-1}{2} + \binom{n-1}{3} = O(n^3)$, where $n := |V|$. Thus, we have concluded that the number of minimum cut-sets in a n-vertex hypergraph is $O(n^3)$.

We recall that the number of minimum cut-sets in a n-vertex hypergraph is known to be at most $\binom{n}{2}$ [34, 38]. So, the $O(n^3)$ upper bound on the number of minimum cut-sets that we obtained above based on Theorem 3.3 appears to be weak. We show the following strengthening of Theorem 3.3 for $k = 2$ to get the tighter bound.

Theorem 3.4. Let $G = (V, E)$ be a hypergraph and $\mathcal{P} = (V_1, V_2)$ be a minimum cut. Then, for all non-empty subsets $T \subseteq V_2$, there exists a subset $S \subseteq V_1$ with $|S| \leq 2$ such that the source minimal minimum (S, T)-terminal cut $(A, \overline{A})$ satisfies $\delta(A) = \delta(V_1)$ and $A \subseteq V_1$.

By applying Theorem 3.4 for $T = \{t\}$, we obtain that for every set $U \in \mathcal{H}$, there exists a subset $S \subseteq U$ with $|S| \leq 2$ such that the source minimal minimum $(S, \{t\})$-terminal cut $(A, \overline{A})$ satisfies $\delta(A) = \delta(U)$. Hence, the size of the set $\mathcal{M}$ is at most the number of possible ways to choose a non-empty subset $S \subseteq V \setminus \{t\}$ of size at most 2 which is $\binom{n-1}{1} + \binom{n-1}{2} = \binom{n}{2}$. Thus, we have obtained a structural explanation (based on Theorem 3.4) for the number of minimum cut-sets in a hypergraph being at most $\binom{n}{2}$. Theorem 3.4 can also be used to enumerate all minimum cut-sets in a given hypergraph using $\binom{n}{2}$ source minimal minimum (S, T)-terminal cut computations.

Theorem 3.4 should be compared with a similar-looking structural theorem for graphs that was shown by Goemans and Ramakrishnan [115]. Goemans and Ramakrishnan showed that (Theorem 15 in [115]) if G is a connected graph, then for every set $U \in \mathcal{H}$, there exists a subset $S \subseteq V_1$ with $|S| \leq 2$ such that $(U, \overline{U})$ is the source minimal minimum $(S, \{t\})$-terminal cut. This leads to a structural explanation for the number of minimum cuts in a connected graph being at most $\binom{n}{2}$. Our Theorem 3.4 can be seen as a counterpart of Goemans and Ramakrishnan's result for hypergraphs, but it differs from their result in two aspects: (1) their result does not hold for hypergraphs—the number of minimum cuts in a connected hypergraph could be exponential as we have seen from the spanning-hyperedge-example and (2) the proof of their result is based on the *submodular triple inequality* which holds only for the graph cut function but fails for the hypergraph cut function. So, our Theorem 3.4 is more general as it handles minimum cut-sets in hypergraphs and moreover, needs a different proof technique compared to [115]. We mention that Goemans and Ramakrishnan's result for connected graphs was our inspiration for Theorem 3.4, which in turn, was our starting point for Theorem 3.3.

We conclude this section with a table summarizing known techniques for solving enumeration problems in graphs and hypergraphs in polynomial time (see Table 3.1). We refer the reader to Section 3.1.2 for a more elaborate discussion of related work.

Organization. In Section 3.1.2, we discuss special cases of Enum-Hypergraph-k-Cut that have been addressed in the literature. In Section 3.1.3, we recall properties of the hypergraph cut function that will be useful to prove our structural theorems. This section contains a strengthening of a partition uncrossing theorem from [73] whose proof appears in Section 3.7. In Section 3.2, we formally describe and analyze the deterministic polynomial-time algorithm for Enum-Hypergraph-k-Cut that utilizes our two structural theorems (Theorems 3.2 and 3.3). In Section 3.3, we prove Theorem 3.2. In Section 3.4, we prove Theorem 3.3. In Section 3.5, we prove the strengthening of Theorem 3.3 for $k = 2$—namely Theorem 3.4. In Section 3.6, we give an alternative proof of Theorem 3.4 based on a novel three-cut-set lemma. In Section 3.7, we prove Theorem 3.5. In Section 3.8, we conclude with

Techniques	**Graph Min-Cut**	**Hypergraph Min-Cut**	**Graph-k-cut**	**Hypergraph-k-cut**
Random Contraction	[16, 65]	[34, 35, 36]	[65, 66, 67, 116]	[35, 36]
Cactus Representation	[15]	[37, 38, 117]	?	?
Min (S,T)-cuts	[115]	**Our Work** (Thm 3.4)	[68, 70]	**Our Work** (Thms 3.2 and 3.3)
Edge Splitting	[118, 119]	?	?	?
Tree Packing	[21]	?	[71, 106]	?

Table 3.1: Summary of known techniques to solve enumeration problems in graphs and hypergraphs in polynomial time. The random contraction technique leads to a randomized algorithm while techniques listed in the rest of the rows lead to deterministic algorithms. Entries with a question mark indicate that we do not *yet* know if the technique leads to a polynomial time algorithm for the corresponding enumeration problem.

a few open problems.

3.1.2 Related Work

In this section, we discuss known techniques for the enumeration problem in the special case of $k = 2$ and the special case of graphs along with challenges involved in adapting these techniques to hypergraphs for $k \geq 3$.

Graph min-k-cut enumeration for $k = 2$. Graph-k-Cut for $k = 2$ is the global minimum cut problem (denoted Graph-MinCut) which has been extensively studied. However, efficient deterministic enumeration of all minimum cut-sets in a given connected graph is already non-trivial. Dinitz, Karzanov, and Lomonosov [15] showed that the number of minimum cuts in a connected graph is at most $\binom{n}{2}$, where n is the number of vertices in the input graph, and they can all be enumerated in deterministic polynomial time. In particular, they designed a compact data structure, namely a cactus graph, to represent all minimum cuts in a connected graph. The upper bound of $\binom{n}{2}$ on the number of minimum cuts in a connected graph is tight as illustrated by the cycle-graph on n vertices. Using the seminal random contraction technique, Karger [16] showed a stronger result that the number of α-approximate minimum cuts in a connected graph is $O(n^{2\alpha})$ and they can all be enumerated in randomized polynomial time for constant α. Karger's tree packing technique [21] also leads to a deterministic polynomial-time algorithm to enumerate all α-approximate minimum cuts in a connected graph for constant α. Nagamochi, Nishimura, and Ibaraki

[119] tightened Karger's bound for a particular value of α via the edge splitting operation: the number of $(4/3-\epsilon)$-approximate minimum cuts in a connected graph is at most $\binom{n}{2}$ for every $\epsilon > 0$. This fact was also shown by Goemans and Ramakrishnan [115] via a structural result (see discussion after Theorem 3.4 above). Henzinger and Williamson [118] extended Nagamochi, Nishimura, and Ibaraki's edge splitting technique to show that the number of $(3/2-\epsilon)$-approximate minimum cuts in a connected graph is $O(n^2)$ for every $\epsilon > 0$. The results of Nagamochi, Nishimura, and Ibaraki, Goemans and Ramakrishnan, and Henzinger and Williamson are all constructive and deterministic (i.e., lead to deterministic polynomial-time algorithms to enumerate the respective approximate minimum cuts) and they bound the number of minimum cuts in a connected graph (as opposed to minimum cut-sets).

Polynomial-delay algorithms. An alternative line of work aims to enumerate *all* cuts in hypergraphs in non-decreasing order of cut value with polynomial time delay between outputs. Such algorithms are known as *polynomial-delay* algorithms in the literature. Polynomial-delay algorithms have been designed based on polynomial-time solvability of minimum (s,t)-terminal cut and using the Lawler-Murty schema [81, 120, 121, 122]. Since we know that the number of minimum cuts in a connected graph is polynomial, the existence of a polynomial-delay algorithm immediately implies a polynomial-time algorithm to solve Enum-Graph-k-Cut for $k=2$. This approach does not extend to Enum-Hypergraph-k-Cut for $k=2$ since the number of minimum cuts in a hypergraph can be exponential (e.g., recall the spanning-hyperedge-example).

Hypergraph min-k-cut-set enumeration for $k=2$. Hypergraph-k-Cut for $k=2$ is the global minimum cut problem (denoted Hypergraph-MinCut) which has also been extensively studied. We note that the number of minimum cuts in a connected hypergraph could be exponential (e.g., consider the spanning-hyperedge-example). But, how about the number of minimum cut-sets? The number of minimum cut-sets in a hypergraph is at most $\binom{n}{2}$ via decomposition theorems of Cunningham and Edmonds [123], Fujishige [124], and Cunningham [117] on submodular functions. Cheng [37] designed an explicit *hypercactus representation* for all minimum cut-sets in a hypergraph. Chekuri and Xu [38] designed a faster deterministic polynomial-time algorithm to obtain a hypercactus representation (along with all minimum cut-sets) of a given hypergraph. Ghaffari, Karger, and Panigrahi [34] (also see [35, 36]) introduced a random contraction technique to solve Hypergraph-MinCut which also implied that the number of minimum cut-sets in a hypergraph is at most $\binom{n}{2}$ and that they can all be enumerated in randomized polynomial time.

We mention that in contrast to graphs, the number of constant-approximate minimum cut-sets in a hypergraph can be exponential. In fact, the number of $(1+\epsilon)$-approximate minimum cut-sets in a connected hypergraph can be exponential[5] for every $\epsilon \in (0,1)$. Moreover, the techniques of Nagamochi, Nishimura, and Ibaraki, Goemans and Ramakrishnan, and Henzinger and Williamson even when restricted to minimum cuts (as opposed to approximate minimum cuts) cannot extend to hypergraphs: This is because, their techniques are tailored to enumerate all minimum cuts in a connected graph as opposed to all minimum cut-sets; we have already seen that the spanning-hyperedge-example has exponential number of minimum cuts and hence, all of them cannot be enumerated in polynomial time.

Multiterminal variants for k-cut: We mention that Graph-k-Cut and Hypergraph-k-Cut have natural variants involving separating specified terminal vertices $s_1, s_2, \ldots, s_k$. These variants are NP-hard for $k \geq 3$ even in graphs and hence, these variants are not viable lines of attack for Graph-k-Cut and Hypergraph-k-Cut. We refer the reader to [73] for a discussion of approximation algorithms for these variants.

Graph min-k-cut enumeration. Graph-k-Cut for $k \geq 3$ has a rich literature with substantial recent work [65, 66, 67, 68, 71, 72, 106, 116, 125, 126, 127, 128, 129, 130]. Goldschmidt and Hochbaum (1988) [72] initiated the study on Graph-k-Cut by showing that it is NP-hard when k is part of the input and that it is polynomial-time solvable when k is a fixed constant (polynomial-time solvability is not obvious even for $k = 3$). Recall that we consider k to be a fixed constant throughout this chapter. Goldschmidt and Hochbaum introduced a divide-and-conquer approach for Graph-k-Cut which resulted in a deterministic polynomial-time algorithm. However, their result did not guarantee a bound on the number of minimum k-partitions or minimum k-cut-sets in connected graphs. Karger and Stein [65] gave a randomized polynomial-time algorithm for Graph-k-Cut via the random contraction technique. In addition, they showed that the number of minimum k-partitions in a connected graph is $O(n^{2k-2})$ and they can all be enumerated in randomized polynomial time. The bound on the number of minimum k-partitions in a connected graph has recently been improved to $O(n^k)$ [66, 67]. We mention that the upper bound of $O(n^k)$ on the number of minimum k-partitions in a connected graph is tight as illustrated by the cycle-graph on n vertices.

There are two known approaches to solve Enum-Graph-k-Cut in deterministic polynomial time: (1) Thorup [106] showed that the tree packing approach can be used to obtain a polynomial-time algorithm for Graph-k-Cut; this approach also extends to solve Enum-Graph-k-Cut (also see [71]). (2) Kamidoi, Yoshida, and Nagamochi [68] extended Goldschmidt and Hochbaum's divide and conquer approach to solve Enum-Graph-k-Cut

[5]Consider the n-vertex hypergraph $G = (V, E)$ where E consists of all size-2 hyperedges each of cost $\delta = \epsilon(\binom{n}{2} - (1+\epsilon)(n-1))^{-1}$ and a hyperedge $e = V$ of cost 1. The cost of a minimum cut is $\lambda := 1 + \delta(n-1)$. The cost of every cut is at most $1 + \delta\binom{n}{2} \leq (1+\epsilon)\lambda$.

(also see [70]).

Hypergraph-k-Cut. The complexity of Hypergraph-k-Cut was open since the work of Goldschmidt and Hochbaum for Graph-k-Cut (1988) [72] until recently. Although certain special cases of Hypergraph-k-Cut were known to be solvable in polynomial time [107, 131], considerable progress on Hypergraph-k-Cut happened only in the last 3 years. Chandrasekaran, Xu, and Yu (2018) [35] designed the first randomized polynomial-time algorithm for Hypergraph-k-Cut; their Monte Carlo algorithm runs in $\tilde{O}(pn^{2k-1})$ time where $p = \sum_{e\in E} |e|$ is the representation size of the input hypergraph. Fox, Panigrahi, and Zhang [36] improved the randomized run-time to $\tilde{O}(mn^{2k-2})$, where m is the number of hyperedges in the input hypergraph. Both these randomized algorithms are based on random contraction of hyperedges and are inspired partly by earlier work in [34] for Hypergraph-MinCut. These randomized algorithms also imply that the number of minimum k-cut-sets is $O(n^{2k-2})$ and that all of them can be enumerated in randomized polynomial time. Chandrasekaran and Chekuri (2020) [73] designed a deterministic polynomial-time algorithm for Hypergraph-k-Cut via a divide and conquer approach. We emphasize that their algorithm finds a minimum k-partition and did not have the tools to find all minimum k-cut-sets.

A polynomial bound on the number of minimum k-cut-sets along with the existence of a randomized polynomial-time algorithm to enumerate all of them raises the possibility of a deterministic algorithm for Enum-Hypergraph-k-Cut. As we mentioned earlier, there are two deterministic approaches for Enum-Graph-k-Cut—tree packing and divide-and-conquer. The tree packing approach does not seem to extend to hypergraphs (even for Hypergraph-MinCut). This leaves the divide-and-conquer approach. Notably, this approach also led to the first deterministic algorithm for Hypergraph-k-Cut in the work of Chandrasekaran and Chekuri [73]. As mentioned earlier, we adapt Chandrasekaran and Chekuri's divide-and-conquer approach and augment it with structural results for minimum k-cut-sets to prove our main result stated in Theorem 3.1.

3.1.3 Preliminaries

Let $G = (V, E)$ be a hypergraph. Throughout, we will follow the notation mentioned in the second paragraph of Section 3.1.1. We will repeatedly rely on the fact that the hypergraph cut function $d : 2^V \to \mathbb{R}_+$ is symmetric and submodular. We recall that a set function $f : 2^V \to \mathbb{R}$ is *symmetric* if $f(U) = f(\overline{U})$ for all $U \subseteq V$ and is *submodular* if $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$ for all subsets $A, B \subseteq V$.

We will need a partition uncrossing theorem that is a strengthening of a result from [73]. We state the strengthened version below. See Figure 3.1 for an illustration of the sets that appear in the statement of Theorem 3.5. We emphasize that the second conclusion in the statement of Theorem 3.5 is the strengthening. The proof of the second conclusion is similar to the proof of the first conclusion which appears in [73]—we present a proof of both conclusions for the sake of completeness in Section 3.7.

Theorem 3.5. Let $G = (V, E)$ be a hypergraph, $k \geq 2$ be an integer and $\emptyset \neq R \subsetneq U \subsetneq V$. Let $S = \{u_1, \ldots, u_p\} \subseteq U \setminus R$ for $p \geq 2k-2$. Let $(\overline{A_i}, A_i)$ be a minimum $((S \cup R) \setminus \{u_i\}, \overline{U})$-terminal cut. Suppose that $u_i \in A_i \setminus (\cup_{j \in [p] \setminus \{i\}} A_j)$ for every $i \in [p]$. Then, the following two hold:

1. There exists a k-partition $(P_1, \ldots, P_k)$ of V with $\overline{U} \subsetneq P_k$ such that
$$\text{cost}(P_1, \ldots, P_k) \leq \frac{1}{2} \min\{d(A_i) + d(A_j) : i, j \in [p], i \neq j\}.$$
2. Moreover, if there exists a hyperedge $e \in E$ such that e intersects $W := \cup_{1 \leq i < j \leq p}(A_i \cap A_j)$, e intersects $Z := \cap_{i \in [p]} \overline{A_i}$, and e is contained in $W \cup Z$, then the inequality in the previous conclusion is strict.

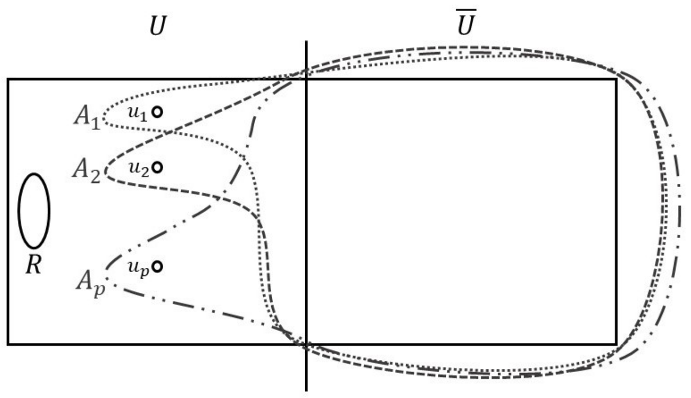

Figure 3.1: Illustration of the sets that appear in the statement of Theorem 3.5.

3.2 ENUMERATION ALGORITHM

We will use Theorems 3.2 and 3.3 to design a deterministic polynomial-time algorithm for Enum-Hypergraph-k-Cut in this section. We describe the formal algorithm in Figure 3.2. It enumerates $n^{O(k)}$ source minimal minimum (S, T)-terminal cuts and considers the cut-set crossing each cut in this collection. If the removal of the cut-set leads to at least k connected components, then it adds such a cut-set to the candidate family $\mathcal{F}$; otherwise, it adds the source set of the cut into a candidate collection $\mathcal{C}$. Next, the algorithm considers all possible k-partitions that can be formed using the sets in the collection $\mathcal{C}$ and adds the set of hyperedges crossing the k-partition to the family $\mathcal{F}$. Finally, it prunes the family $\mathcal{F}$ to return all minimum k-cut-sets in it. The run-time guarantee and the cardinality of the family of k-cut-sets returned by the algorithm are given in Theorem 3.6. Theorem 3.1 follows from Theorem 3.6 by observing that the source minimal minimum (S,T)-terminal cut in a hypergraph can be computed in deterministic polynomial time—e.g., it can be computed in a n-vertex hypergraph of size p in $O(np)$ time [38].

```
Algorithm Enum-Cuts($G = (V, E), k$)
  Input: Hypergraph $G = (V, E)$ and an integer $k \geq 2$
  Output: Family of all minimum $k$-cut-sets in $G$
  Initialize $\mathcal{C} \leftarrow \emptyset$, $\mathcal{F} \leftarrow \emptyset$
  For each pair $(S, T)$ such that $S, T \subseteq V$ with $S \cap T = \emptyset$ and $|S|, |T| \leq 2k - 1$
      Compute the source minimal minimum $(S, T)$-terminal cut $(U, \overline{U})$
      If $G - \delta(U)$ has at least $k$ connected components
          $\mathcal{F} \leftarrow \mathcal{F} \cup \{\delta(U)\}$
      Else
          $\mathcal{C} \leftarrow \mathcal{C} \cup \{U\}$
  For each $k$-partition $(U_1, \ldots, U_k)$ of $V$ with $U_1, \ldots, U_k \in \mathcal{C}$
      $\mathcal{F} \leftarrow \mathcal{F} \cup \{\delta(U_1, \ldots, U_k)\}$
  Among all $k$-cut-sets in the family $\mathcal{F}$, return the subfamily of cheapest ones
```

Figure 3.2: Algorithm to enumerate hypergraph minimum k-cut-sets

Theorem 3.6. Let $G = (V, E)$ be a n-vertex hypergraph of size p and let k be an integer. Then, Algorithm Enum-Cuts(G, k) in Figure 3.2 returns the family of all minimum k-cut-sets in G and it can be implemented to run in $O(n^{4k-2})T(n,p) + O(n^{4k^2-2k}p)$ time, where $T(n,p)$ denotes the time complexity for computing the source minimal minimum (s,t)-terminal cut in a n-vertex hypergraph of size p. Moreover, the cardinality of the family returned by the algorithm is $O(n^{2k(2k-1)})$.

Proof. We begin by showing correctness. The last step of the algorithm considers only k-cut-sets in the family $\mathcal{F}$, so the algorithm returns a subfamily of k-cut-sets. We only have to show that every minimum k-cut-set is in the family $\mathcal{F}$; this will also guarantee that every k-cut-set in the returned subfamily is indeed a minimum k-cut-set.

Let $F \subseteq E$ be a minimum k-cut-set in G and let $(V_1, \ldots, V_k)$ be a minimum k-partition such that $F = \delta(V_1, \ldots, V_k)$. We will show that F is in the family $\mathcal{F}$. We know that $d(V_i) \leq OPT_k$ for every $i \in [k]$. We distinguish two cases:

1. Suppose $d(V_i) < OPT_k$ for every $i \in [k]$.

   Consider an arbitrary part V_i where $i \in [k]$. By Theorem 3.2, there exist disjoint subsets $S, T \subseteq V$ with $|S|, |T| \leq 2k - 2$ such that $(V_i, \overline{V_i})$ is the unique minimum (S, T)-terminal cut. Hence, the set V_i is in the collection $\mathcal{C}$. Consequently, all parts $V_1, \ldots, V_k$ are in the collection $\mathcal{C}$. Hence, the set $F = \delta(V_1, \ldots, V_k)$ is added to the family $\mathcal{F}$ in the second for-loop.

2. Suppose there exists $i \in [k]$ such that $d(V_i) = OPT_k$.

   In this case, we have $\delta(V_i) = F = \delta(V_1, \ldots, V_k)$. By Theorem 3.3, there exist disjoint subsets $S, T \subseteq V$ with $|S|, |T| \leq 2k - 1$ such that the source minimal minimum (S, T)-terminal cut $(A, \overline{A})$ satisfies $\delta(A) = \delta(V_i) = F$. Therefore, the set F is added to the family $\mathcal{F}$ in the first for-loop.

Thus, in both cases, we have shown that the set F is contained in the family $\mathcal{F}$. Since the algorithm returns the subfamily of hyperedge sets in $\mathcal{F}$ that correspond to minimum k-cut-sets, the set F is in the family returned by the algorithm.

Next, we bound the run time and the number of minimum k-cut-sets returned by the algorithm. The first for-loop can be implemented using $O(n^{4k-2})$ source minimal minimum (s, t)-terminal cut computations. Moreover, the size of the collection $\mathcal{C}$ is $O(n^{4k-2})$. The number of tuples $(U_1, \ldots, U_k) \in \mathcal{C}^k$ is $O(n^{4k^2-2k})$. Verifying if a tuple $(U_1, \ldots, U_k)$ forms a k-partition takes $O(n)$ time. For a tuple which forms a k-partition, computing the hyperedges crossing that partition takes $O(p)$ time. Thus, the second for-loop can be implemented to run in time $O(n^{4k^2-2k}p)$. The size of the family $\mathcal{F}$ is $O(n^{4k^2-2k})$. Each k-cut-set in $\mathcal{F}$ has representation size at most p. Hence, computing the size of each k-cut-set in $\mathcal{F}$ and returning the cheapest ones can be implemented to run in time $O(n^{4k^2-2k}p)$. Thus, the overall run-time is $O(n^{4k-2})T(n, p) + O(n^{4k^2-2k}p)$. QED.

3.3 PROOF OF THEOREM 3.2

We prove Theorem 3.2 in this section. We will use the following theorem to prove Theorem 3.2.

Theorem 3.7. Let $G = (V, E)$ be a hypergraph and let OPT_k be the value of a minimum k-cut-set in G for some integer $k \geq 2$. Suppose $(U, \overline{U})$ is a 2-partition of V with $d(U) < OPT_k$. Then, for every vertex $s \in U$, there exists a subset $S \subseteq U \backslash \{s\}$ with $|S| \leq 2k - 3$ such that $(U, \overline{U})$ is the unique minimum $(S \cup \{s\}, \overline{U})$-terminal cut.

Proof. Let $s \in U$. Consider the collection

$$\mathcal{C} := \{Q \subseteq V \backslash \{s\} : \overline{U} \subsetneq Q, d(Q) \leq d(U)\}. \tag{3.3}$$

Let S be an inclusion-wise minimal subset of $U \setminus \{s\}$ such that $S \cap Q \neq \emptyset$ for all $Q \in \mathcal{C}$, i.e., the set S is completely contained in $U \backslash \{s\}$ and is a minimal transversal of $\mathcal{C}$. Proposition 3.1 and Lemma 3.1 complete the proof of Theorem 3.7 for this choice of S. QED.

Proposition 3.1. The 2-partition $(U, \overline{U})$ is the unique minimum $(S \cup \{s\}, \overline{U})$-terminal cut.

Proof. For the sake of contradiction, suppose $(Y, \overline{Y})$ is a minimum $(S \cup \{s\}, \overline{U})$-terminal cut with $Y \neq U$. This implies that $S \cup \{s\} \subseteq Y$ and $\overline{U} \subsetneq \overline{Y}$. Moreover, we have $d(\overline{Y}) \leq d(\overline{U})$ because $(U, \overline{U})$ is a $(S \cup \{s\}, \overline{U})$-terminal cut. Consequently, the set $\overline{Y}$ is in the collection $\mathcal{C}$. Since S is a transversal of the collection $\mathcal{C}$, we have that $S \cap \overline{Y} \neq \emptyset$. This contradicts the fact that S is contained in Y. QED.

Lemma 3.1. The size of the subset S is at most $2k - 3$.

Proof. For the sake of contradiction, suppose $|S| \geq 2k - 2$. Our proof strategy is to show the existence of a k-partition with cost smaller than OPT_k, thus contradicting the definition of OPT_k. Let $S := \{u_1, u_2, \ldots, u_p\}$ for some $p \geq 2k - 2$. For each $i \in [p]$, let $(\overline{A_i}, A_i)$ be the source minimal minimum $((S \cup \{s\}) \backslash \{u_i\}, \overline{U})$-terminal cut. The following claim will allow us to show that the cuts $(\overline{A_i}, A_i)$ satisfy the hypothesis of Theorem 3.5.

Claim 3.1. For every $i \in [p]$, we have $d(A_i) \leq d(U)$ and $u_i \in A_i$.

Proof. Let $i \in [p]$. Since S is a minimal transversal of the collection $\mathcal{C}$, there exists a set $B_i \in \mathcal{C}$ such that $B_i \cap S = \{u_i\}$. Hence, $(\overline{B_i}, B_i)$ is a $((S \cup \{s\}) \backslash \{u_i\}, \overline{U})$-terminal cut. Therefore,

$$d(A_i) \leq d(B_i) \leq d(U). \tag{3.4}$$

We will show that A_i is in the collection $\mathcal{C}$. By definition, $A_i \subseteq V \setminus \{s\}$ and $\overline{U} \subseteq A_i$. If $A_i = \overline{U}$, then the above inequalities are equations implying that $(B_i, \overline{B_i})$ is a minimum $((S \cup \{s\}) \backslash \{u_i\}, \overline{U})$-terminal cut, and consequently, $(B_i, \overline{B_i})$ contradicts source minimality of the minimum $((S \cup \{s\}) \backslash \{u_i\}, \overline{U})$-terminal cut $(A_i, \overline{A_i})$. Therefore, $\overline{U} \subsetneq A_i$. Hence, A_i is in the collection $\mathcal{C}$.

We recall that the set S is a transversal for the collection $\mathcal{C}$ and moreover, none of the elements of $S \setminus \{u_i\}$ are in A_i by definition of A_i. Therefore, the vertex u_i must be in A_i. QED.

Using Claim 3.1, we observe that the sets U, $R := \{s\}$, S, and the partitions $(\overline{A_i}, A_i)$ for $i \in [p]$ satisfy the conditions of Theorem 3.5. By the first conclusion of Theorem 3.5 and Claim 3.1, we obtain a k-partition $(P_1, \dots, P_k)$ of V such that

$$\text{cost}(P_1, \dots, P_k) \le \frac{1}{2} \min\{d(A_i) + d(A_j) : i, j \in [p], i \neq j\} \le d(U) < OPT_k. \tag{3.5}$$

The last inequality above is by the assumption in the theorem statement. Thus, we have obtained a k-partition whose cost is smaller than OPT_k, a contradiction.

QED.

Applying Theorem 3.7 to $(\overline{U}, U)$ yields the following corollary.

Corollary 3.1. Let $G = (V, E)$ be a hypergraph and let OPT_k be the value of a minimum k-cut-set in G for some integer $k \ge 2$. Suppose $(U, \overline{U})$ is a 2-partition of V with $d(U) < OPT_k$. Then, for every vertex $t \in \overline{U}$, there exists a subset $T \subseteq \overline{U} \backslash \{t\}$ with $|T| \le 2k - 3$ such that $(U, \overline{U})$ is the unique minimum $(U, T \cup \{t\})$-terminal cut.

We now restate Theorem 3.2 and prove it using Theorem 3.7 and Corollary 3.1.

Theorem 3.2. Let $G = (V, E)$ be a hypergraph and let OPT_k be the value of a minimum k-cut-set in G for some integer $k \ge 2$. Suppose $(U, \overline{U})$ is a 2-partition of V with $d(U) < OPT_k$. Then, for every pair of vertices $s \in U$ and $t \in \overline{U}$, there exist subsets $S \subseteq U \setminus \{s\}$ and $T \subseteq \overline{U} \setminus \{t\}$ with $|S| \le 2k - 3$ and $|T| \le 2k - 3$ such that $(U, \overline{U})$ is the unique minimum $(S \cup \{s\}, T \cup \{t\})$-terminal cut in G.

Proof. Let $s \in U$ and $t \in \overline{U}$. By Theorem 3.7, there exists a subset $S \subseteq U \setminus \{s\}$ such that $|S| \le 2k - 3$ and $(U, \overline{U})$ is the unique minimum $(S \cup \{s\}, \overline{U})$-terminal cut. By Corollary 3.1, there exists a subset $T \subseteq \overline{U}$ such that $|T| \le 2k - 3$ and $(U, \overline{U})$ is the unique minimum $(U, T \cup \{t\})$-terminal cut.

We now show that $(U, \overline{U})$ is the unique minimum $(S \cup \{s\}, T \cup \{t\})$-terminal cut. Let $(Y, \overline{Y})$ be a minimum $(S \cup \{s\}, T \cup \{t\})$-terminal cut. Suppose $Y \neq U$. We have the following observations:

1. Since $(U, \overline{U})$ is a $(S \cup \{s\}, T \cup \{t\})$-terminal cut, we have that $d(U) \ge d(Y)$.
2. Since $(U \cap Y, \overline{U \cap Y})$ is a $(S \cup \{s\}, \overline{U})$-terminal cut, we have that $d(U \cap Y) \ge d(U)$.
3. Since $(U \cup Y, \overline{U \cup Y})$ is a $(U, T \cup \{t\})$-terminal cut, we have that $d(U \cup Y) \ge d(U)$.

Moreover, since $Y \neq U$, we have that either $U \cap Y \neq U$ or $U \cup Y \neq U$. Since $(U, \overline{U})$ is the unique minimum $(S \cup \{s\}, \overline{U})$-terminal cut and also the unique minimum $(U, T \cup \{t\})$-terminal cut, it follows that either $d(U \cap Y) > d(U)$ or $d(U \cup Y) > d(U)$. These observations in conjunction with the submodularity of the hypergraph cut function imply that

$$2d(U) \geq d(U) + d(Y) \geq d(U \cap Y) + d(U \cup Y) > 2d(U), \tag{3.6}$$

a contradiction. Hence, $Y = U$. QED.

3.4 PROOF OF THEOREM 3.3

We prove Theorem 3.3 in this section. We begin with the following useful containment lemma. Variants of this containment lemma have appeared in the literature before under slightly different hypothesis (e.g., see [72, 73, 114, 132]).

Lemma 3.2. Let $G = (V, E)$ be a hypergraph, $k \geq 2$ be an integer, $\mathcal{P} = (V_1, \ldots, V_k)$ be a minimum k-partition such that $\delta(\mathcal{P}) = \delta(V_1)$, and $S \subseteq V_1$, $T \subseteq \overline{V_1}$ such that $T \cap V_j \neq \emptyset$ for all $j \in \{2, 3, \ldots, k\}$. Suppose that $(U, \overline{U})$ is the source minimal minimum (S, T)-terminal cut. Then, $U \subseteq V_1$ and $(U, \overline{U})$ is a minimum $(S, \overline{V_1})$-terminal cut.

Proof. We note that $S \subseteq U \cap V_1$, so $(U \cap V_1, \overline{U \cap V_1})$ is a (S, T)-terminal cut. Thus, we have

$$d(U \cap V_1) \geq d(U). \tag{3.7}$$

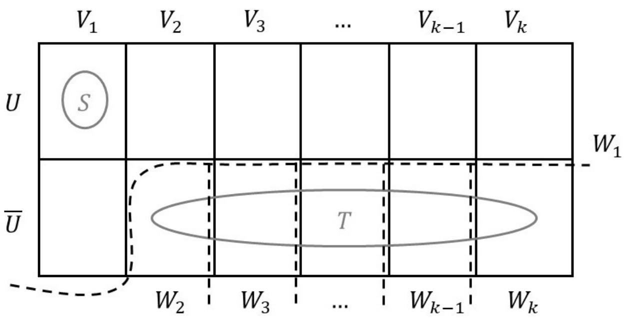

Figure 3.3: Uncrossing in the proof of Lemma 3.2.

Now consider $\mathcal{P}' = (W_1 := U \cup V_1, W_2 := V_2 \setminus U, \ldots, W_k := V_k \setminus U)$ (see Figure 3.3). For each $i \in \{2, 3, \ldots, k\}$, we have $\emptyset \neq T \cap V_i \subseteq V_i \setminus U$, so $\mathcal{P}'$ is a k-partition. Since $\delta(\mathcal{P}) = \delta(V_1)$, every hyperedge which crosses $\mathcal{P}$ must intersect V_1. Consequently, every hyperedge which crosses $\mathcal{P}'$ must intersect $U \cup V_1$. Therefore

$$d(U \cup V_1) = \text{cost}(\mathcal{P}') \geq \text{cost}(\mathcal{P}) = d(V_1). \tag{3.8}$$

By submodularity of the hypergraph cut function and inequalities (3.7) and (3.8), we have that

$$d(U) + d(V_1) \geq d(U \cap V_1) + d(U \cup V_1) \geq d(U) + d(V_1). \tag{3.9}$$

Therefore, inequality (3.7) is in fact an equation and hence, $(U \cap V_1, \overline{U \cap V_1})$ is a minimum (S, T)-terminal cut. If $U \setminus V_1 \neq \emptyset$, then $(U \cap V_1, \overline{U \cap V_1})$ contradicts source minimality of the minimum (S, T)-terminal cut $(U, \overline{U})$. Hence, $U \setminus V_1 = \emptyset$ and consequently, $U \subseteq V_1$.

Since $U \subseteq V_1$, we have that $(U, \overline{U})$ is a $(S, \overline{V_1})$-terminal cut. Furthermore, since $T \subseteq \overline{V_1}$, every $(S, \overline{V_1})$-terminal cut is also a (S, T)-terminal cut. Therefore, every $(S, \overline{V_1})$-terminal cut must have weight at least $d(U)$, and hence $(U, \overline{U})$ is a minimum $(S, \overline{V_1})$-terminal cut. QED.

We now restate and prove Theorem 3.3.

Theorem 3.3. Let $G = (V, E)$ be a hypergraph, $k \geq 2$ be an integer, and $\mathcal{P} = (V_1, \ldots, V_k)$ be a minimum k-partition such that $\delta(V_1) = \delta(\mathcal{P})$. Then, for all subsets $T \subseteq \overline{V_1}$ such that $T \cap V_j \neq \emptyset$ for all $j \in \{2, 3, \ldots, k\}$, there exists a subset $S \subseteq V_1$ with $|S| \leq 2k - 1$ such that the source minimal minimum (S, T)-terminal cut $(A, \overline{A})$ satisfies $\delta(A) = \delta(V_1)$ and $A \subseteq V_1$.

Proof. Let us fix an arbitrary $T \subseteq \overline{V_1}$ such that $T \cap V_j \neq \emptyset$ for all $j \in \{2, \ldots, k\}$. For a subset $X \subseteq V_1$, we denote the source minimal minimum (X, T)-terminal cut by $(H_X, \overline{H_X})$. By Lemma 3.2, for all $X \subseteq V_1$ we have that $H_X \subseteq V_1$ and $d(H_X) \leq d(V_1)$. If $|V_1| \leq 2k - 1$, then choosing $S = V_1$ proves the theorem. So, we will assume henceforth that $|V_1| > 2k - 1$. We will show that there exists a subset $S \subseteq V_1$ with $|S| \leq 2k - 1$ such that the source minimal minimum (S, T)-terminal cut $(H_S, \overline{H_S})$ satisfies $\delta(H_S) = \delta(V_1)$. This suffices since we have that $H_S \subseteq V_1$ for all subsets $S \subseteq V_1$ (by Lemma 3.2).

For the sake of contradiction, suppose that for every $S \subseteq V_1$ with $|S| \leq 2k - 1$, the source minimal minimum (S, T)-terminal cut $(H_S, \overline{H_S})$ does not satisfy $\delta(H_S) = \delta(V_1)$. Our proof strategy is to obtain a cheaper k-partition than $(V_1, \ldots, V_k)$, thereby contradicting the optimality of $(V_1, \ldots, V_k)$.

Let $S \subseteq V_1$ be a set of size $2k-1$ such that H_S is maximal—i.e., there does not exist $S' \subseteq V_1$ of size $2k - 1$ such that $H_{S'} \supsetneq H_S$. Let $S := \{u_1, u_2, \ldots, u_{2k-1}\}$. By assumption, we have that $\delta(H_S) \neq \delta(V_1)$, but since $(V_1, \overline{V_1})$ is a (S, T)-terminal cut, we have that $d(H_S) \leq d(V_1)$. Therefore, $\delta(V_1) \setminus \delta(H_S)$ is non-empty. Let us fix a hyperedge $e \in \delta(V_1) \setminus \delta(H_S)$. Let $u_{2k} \in e \cap V_1$. Let $C := S \cup \{u_{2k}\} = \{u_1, \ldots, u_{2k-1}, u_{2k}\}$. For notational convenience we will use $C - u_i$ to denote $C \setminus \{u_i\}$ and $C - u_i - u_j$ to denote $C \setminus \{u_i, u_j\}$ for all $i, j \in [2k]$. The choice of the hyperedge e is crucial to our proof—its properties will be used much later in our proof. We summarize the properties of the hyperedge e here.

Observation 3.1. The hyperedge e has the following properties:

1. $e \cap \overline{V_1} \neq \emptyset$,

2. $u_{2k} \in e$, and

3. $e \subseteq \overline{H_S}$.

Our strategy to arrive at a cheaper k-partition than $(V_1, \ldots, V_k)$ is to apply the second conclusion of Theorem 3.5. The next few claims will set us up to obtain sets that satisfy the hypothesis of Theorem 3.5.

Claim 3.2. For every $i \in [2k]$, we have $u_i \notin H_{C-u_i}$.

Proof. If $i = 2k$, then by Observation 3.1 we have $u_{2k} \in e$ and $e \subseteq \overline{H_S}$ so $u_{2k} \notin H_S = H_{C-u_{2k}}$. Suppose $i \in [2k-1]$. Our proof will rely on the choice of S.

Suppose for contradiction that $u_i \in H_{C-u_i}$ for some $i \in [2k-1]$. Then, we have that $S \subseteq H_{C-u_i}$, so $(H_{C-u_i} \cap H_S, \overline{H_{C-u_i} \cap H_S})$ is a (S,T)-terminal cut. Therefore,

$$d(H_{C-u_i} \cap H_S) \geq d(H_S). \tag{3.10}$$

Also, since $(H_{C-u_i} \cup H_S, \overline{H_{C-u_i} \cup H_S})$ is a $(C - u_i, T)$-terminal cut, we have that

$$d(H_{C-u_i} \cup H_S) \geq d(H_{C-u_i}). \tag{3.11}$$

By submodularity of the hypergraph cut function and inequalities (3.10) and (3.11), we have that

$$d(H_S) + d(H_{C-u_i}) \geq d(H_{C-u_i} \cap H_S) + d(H_{C-u_i} \cup H_S) \geq d(H_S) + d(H_{C-u_i}). \tag{3.12}$$

Therefore, inequality (3.10) is an equation, and consequently, $(H_{C-u_i} \cap H_S, \overline{H_{C-u_i} \cap H_S})$ is a minimum (S,T)-terminal cut. If $H_{C-u_i} \cap H_S \subsetneq H_S$, then $(H_{C-u_i} \cap H_S, \overline{H_{C-u_i} \cap H_S})$ contradicts source minimality of the minimum (S,T)-terminal cut $(H_S, \overline{H_S})$. Therefore $H_{C-u_i} \cap H_S = H_S$ and hence, $H_S \subseteq H_{C-u_i}$. Also, the vertex u_{2k} is in $C - u_i$ but not in H_S and hence, $H_S \subsetneq H_{C-u_i}$. However, $|C - u_i| = 2k - 1$. Therefore, the set $C - u_i$ contradicts the choice of S. QED.

The following claim will help in showing that $u_i, u_j \notin H_{C-u_i-u_j}$, which in turn, will be used to show that the hypothesis of Theorem 3.5 is satisfied by suitably chosen sets.

Claim 3.3. For every $i, j \in [2k]$, we have $H_{C-u_i-u_j} \subseteq H_{C-u_i}$.

Proof. We may assume that $i \neq j$. We note that $(H_{C-u_i-u_j} \cap H_{C-u_i}, \overline{H_{C-u_i-u_j} \cap H_{C-u_i}})$ is a $(C - u_i - u_j, T)$-terminal cut. Therefore

$$d(H_{C-u_i-u_j} \cap H_{C-u_i}) \geq d(H_{C-u_i-u_j}). \tag{3.13}$$

Also, $(H_{C-u_i-u_j} \cup H_{C-u_i}, \overline{H_{C-u_i-u_j} \cup H_{C-u_i}})$ is a $(C - u_i, T)$-terminal cut. Therefore

$$d(H_{C-u_i-u_j} \cup H_{C-u_i}) \geq d(H_{C-u_i}). \tag{3.14}$$

By submodularity of the hypergraph cut function and inequalities (3.13) and (3.14), we have that

$$\begin{aligned} d(H_{C-u_i-u_j}) + d(H_{C-u_i}) &\geq d(H_{C-u_i-u_j} \cap H_{C-u_i}) + d(H_{C-u_i-u_j} \cup H_{C-u_i}) &(3.15)\\ &\geq d(H_{C-u_i-u_j}) + d(H_{C-u_i}). &(3.16)\end{aligned}$$

Therefore, inequality (3.13) is an equation, and consequently, we have that $(H_{C-u_i-u_j} \cap H_{C-u_i}, \overline{H_{C-u_i-u_j} \cap H_{C-u_i}})$ is a minimum $(C - u_i - u_j, T)$-terminal cut. If $H_{C-u_i-u_j} \setminus H_{C-u_i} \neq \emptyset$, then $(H_{C-u_i-u_j} \cap H_{C-u_i}, \overline{H_{C-u_i-u_j} \cap H_{C-u_i}})$ contradicts source minimality of the minimum $(C - u_i - u_j, T)$-terminal cut $(H_{C-u_i-u_j}, \overline{H_{C-u_i-u_j}})$. Hence, $H_{C-u_i-u_j} \setminus H_{C-u_i} = \emptyset$ and consequently, $H_{C-u_i-u_j} \subseteq H_{C-u_i}$.

QED.

Claim 3.3 implies the following Corollary.

Corollary 3.2. For every $i \in [2k]$, we have $u_i, u_j \notin H_{C-u_i-u_j}$.

Proof. By Claim 3.2, we have that $u_i \notin H_{C-u_i}$ and $u_j \notin H_{C-u_j}$. Therefore, $u_i, u_j \notin H_{C-u_i} \cap H_{C-u_j}$. By Claim 3.3, $H_{C-u_i-u_j} \subseteq H_{C-u_i}$ and $H_{C-u_i-u_j} \subseteq H_{C-u_j}$. Therefore, $H_{C-u_i-u_j} \subseteq H_{C-u_i} \cap H_{C-u_j}$, and thus, $u_i, u_j \notin H_{C-u_i-u_j}$. QED.

The next claim will help in controlling the cost of the k-partition that we will obtain by applying Theorem 3.5.

Claim 3.4. For every $i, j \in [2k]$, we have $d(H_{C-u_i}) = d(V_1) = d(H_{C-u_i-u_j})$.

Proof. Let $a, b \in [2k]$. Since $(V_1, \overline{V_1})$ is a $(C - u_a, T)$-terminal cut, we have that $d(H_{C-u_a}) \leq d(V_1)$. Since $(H_{C-u_a}, \overline{H_{C-u_a}})$ is a $(C - u_a - u_b, T)$-terminal cut, we have that $d(H_{C-u_a-u_b}) \leq d(H_{C-u_a}) \leq d(V_1)$. Thus, in order to prove the claim, it suffices to show that $d(H_{C-u_a-u_b}) \geq d(V_1)$.

Suppose for contradiction that $d(H_{C-u_a-u_b}) < d(V_1)$. Let $\ell \in [2k] \setminus \{a, b\}$ be an arbitrary element (which exists since $k \geq 2$). Let $R := \{u_\ell\}$, $U := V_1$, $S' := C - u_a - u_\ell$, and $A_i := \overline{H_{C-u_a-u_i}}$ for every $i \in [2k] \setminus \{a, \ell\}$. We note that $|S'| = 2k - 2$. By Lemma 3.2, we have that $(\overline{A_i}, A_i)$ is a minimum $(C - u_a - u_i, \overline{V_1})$-terminal cut for every $i \in [2k] \setminus \{a, \ell\}$. Moreover, by Corollary 3.2, we have that $u_i \in A_i \setminus (\cup_{j \in [2k] \setminus \{a,i,\ell\}} A_j)$ for every $i \in [2k] \setminus \{a, \ell\}$. Hence, the sets U, R, and S', and the cuts $(\overline{A_i}, A_i)$ for $i \in [2k] \setminus \{a, \ell\}$ satisfy the conditions of Theorem 3.5. Therefore, by the first conclusion of Theorem 3.5, there exists a k-partition $\mathcal{P}'$ with

$$\text{cost}(\mathcal{P}') \leq \frac{1}{2} \min\{d(H_{C-u_a-u_i}) + d(H_{C-u_a-u_j})\colon i, j \in [2k] \setminus \{a, \ell\}\}. \tag{3.17}$$

By assumption, $d(H_{C-u_a-u_b}) < d(V_1)$ and $b \in [2k] \setminus \{a, \ell\}$, so $\min\{d(H_{C-u_a-u_i})\colon i \in [2k] \setminus \{a, \ell\}\} < d(V_1)$. Since $(V_1, \overline{V_1})$ is a $(C - u_a - u_i, T)$-terminal cut, we have that $d(H_{C-u_a-u_i}) \leq d(V_1)$ for every $i \in [2k] \setminus \{a, \ell\}$. Therefore,

$$\frac{1}{2} \min\{d(H_{C-u_a-u_i}) + d(H_{C-u_a-u_j})\colon i, j \in [2k] \setminus \{a, \ell\}\} < d(V_1) = \text{cost}(\mathcal{P}). \tag{3.18}$$

Thus, we have that $\text{cost}(\mathcal{P}') < \text{cost}(\mathcal{P})$, which is a contradiction, since $\mathcal{P}$ is a minimum k-partition. QED.

The next two claims will help in arguing properties about the hyperedge e which will allow us to use the second conclusion of Theorem 3.5. In particular, we will need Claim 3.6. The following claim will help in proving Claim 3.6.

Claim 3.5. For every $i, j \in [2k]$, we have

$$d(H_{C-u_i} \cap H_{C-u_j}) = d(V_1) = d(H_{C-u_i} \cup H_{C-u_j}).$$

Proof. Since $(H_{C-u_i} \cap H_{C-u_j}, \overline{H_{C-u_i} \cap H_{C-u_j}})$ is a $(C - u_i - u_j, T)$-terminal cut, we have that $d(H_{C-u_i} \cap H_{C-u_j}) \geq d(H_{C-u_i-u_j})$. By Claim 3.4, we have that $d(H_{C-u_i-u_j}) = d(V_1) = d(H_{C-u_i})$. Therefore,

$$d(H_{C-u_i} \cap H_{C-u_j}) \geq d(H_{C-u_i}). \tag{3.19}$$

Since $(H_{C-u_i} \cup H_{C-u_j}, \overline{H_{C-u_i} \cup H_{C-u_j}})$ is a $(C - u_j, T)$-terminal cut, we have that

$$d(H_{C-u_i} \cup H_{C-u_j}) \geq d(H_{C-u_j}). \tag{3.20}$$

By submodularity of the hypergraph cut function and inequalities (3.19) and (3.20), we have that

$$d(H_{C-u_i})+d(H_{C-u_j}) \geq d(H_{C-u_i} \cap H_{C-u_j})+d(H_{C-u_i} \cup H_{C-u_j}) \geq d(H_{C-u_i})+d(H_{C-u_j}). \quad (3.21)$$

Therefore, inequalities (3.19) and (3.20) are equations. Thus, by Claim 3.4, we have that

$$d(H_{C-u_i} \cap H_{C-u_j}) = d(H_{C-u_i}) = d(V_1), \quad (3.22)$$

and

$$d(H_{C-u_i} \cup H_{C-u_j}) = d(H_{C-u_j}) = d(V_1). \quad (3.23)$$

QED.

Claim 3.6. For every $i, j, \ell \in [2k]$ with $i \neq j$, we have $H_{C-u_\ell} \subseteq H_{C-u_i} \cup H_{C-u_j}$.

Proof. If $\ell = i$ or $\ell = j$ the claim is immediate. Thus, we assume that $\ell \notin \{i, j\}$. Let $Q := H_{C-u_\ell} \setminus (H_{C-u_i} \cup H_{C-u_j})$. We need to show that $Q = \emptyset$. We will show that $(H_{C-u_\ell} \setminus Q, \overline{H_{C-u_\ell} \setminus Q})$ is a minimum $(C - u_\ell, T)$-terminal cut. Consequently, Q must be empty (otherwise, $H_{C-u_\ell} \setminus Q \subsetneq H_{C-u_\ell}$ and hence, $(H_{C-u_\ell} \setminus Q, \overline{H_{C-u_\ell} \setminus Q})$ contradicts source minimality of the minimum $(C - u_\ell, T)$-terminal cut $(H_{C-u_\ell}, \overline{H_{C-u_\ell}})$).

We now show that $(H_{C-u_\ell} \setminus Q, \overline{H_{C-u_\ell} \setminus Q})$ is a minimum $(C - u_\ell, T)$-terminal cut. Since $H_{C-u_\ell} \setminus Q = H_{C-u_\ell} \cap (H_{C-u_i} \cup H_{C-u_j})$, we have that $C - u_i - u_j - u_\ell \subseteq H_{C-u_\ell} \setminus Q$. We also know that u_i and u_j are contained in both H_{C-u_ℓ} and $H_{C-u_i} \cup H_{C-u_j}$. Therefore, $C - u_\ell \subseteq H_{C-u_\ell} \setminus Q$. Thus, $(H_{C-u_\ell} \setminus Q, \overline{H_{C-u_\ell} \setminus Q})$ is a $(C - u_\ell, T)$-terminal cut. Therefore,

$$d(H_{C-u_\ell} \cap (H_{C-u_i} \cup H_{C-u_j})) = d(H_{C-u_\ell} \setminus Q) \geq d(H_{C-u_\ell}). \quad (3.24)$$

We also have that $(H_{C-u_\ell} \cup (H_{C-u_i} \cup H_{C-u_j}), \overline{H_{C-u_\ell} \cup (H_{C-u_i} \cup H_{C-u_j})})$ is a $(C - u_i, T)$-terminal cut. Therefore, $d(H_{C-u_\ell} \cup (H_{C-u_i} \cup H_{C-u_j})) \geq d(H_{C-u_i})$. By Claims 3.4 and 3.5, we have that $d(H_{C-u_i}) = d(V_1) = d(H_{C-u_i} \cup H_{C-u_j})$. Therefore,

$$d(H_{C-u_\ell} \cup (H_{C-u_i} \cup H_{C-u_j})) \geq d(H_{C-u_i} \cup H_{C-u_j}). \quad (3.25)$$

By submodularity of the hypergraph cut function and inequalities (3.24) and (3.25), we have

that

$$d(H_{C-u_\ell}) + d(H_{C-u_i} \cup H_{C-u_j}) \tag{3.26}$$
$$\geq d(H_{C-u_\ell} \cap (H_{C-u_i} \cup H_{C-u_j})) + d(H_{C-u_\ell} \cup (H_{C-u_i} \cup H_{C-u_j})) \tag{3.27}$$
$$\geq d(H_{C-u_\ell}) + d(H_{C-u_i} \cup H_{C-u_j}). \tag{3.28}$$

Therefore, inequalities (3.24) and (3.25) are equations, so $(H_{C-u_\ell} \setminus Q, \overline{H_{C-u_\ell} \setminus Q})$ is a minimum $(C - u_\ell, T)$-terminal cut. QED.

Let $R := \{u_{2k}\}$, $U := V_1$, and let $(\overline{A_i}, A_i) := (H_{C-u_i}, \overline{H_{C-u_i}})$ for every $i \in [2k-1]$. By Lemma 3.2, we have that $(\overline{A_i}, A_i)$ is a minimum $(C-u_i, \overline{V_1})$-terminal cut for every $i \in [2k-1]$. Moreover, by Claim 3.2, we have that $u_i \in A_i \setminus (\cup_{j\in[2k-1]\setminus\{i\}} A_j)$. Hence, the sets U, R, and S, and the cuts $(\overline{A_i}, A_i)$ for $i \in [2k-1]$ satisfy the conditions of Theorem 3.5. We will use the second conclusion of Theorem 3.5. We now show that the hyperedge e that we fixed at the beginning of the proof satisfies the conditions mentioned in the second conclusion of Theorem 3.5. We will use Claim 3.6 to prove this. Let $W := \cup_{1\leq i<j\leq 2k-1}(A_i \cap A_j)$ and $Z := \cap_{i\in[2k-1]}\overline{A_i}$ as in the statement of Theorem 3.5.

Claim 3.7. The hyperedge e satisfies the following conditions:

1. $e \cap W \neq \emptyset$,
2. $e \cap Z \neq \emptyset$, and
3. $e \subseteq W \cup Z$.

Proof. 1. By Lemma 3.2, for every $i \in [2k-1]$ we have $\overline{V_1} \subseteq A_i$, and therefore $\overline{V_1} \subseteq W$. Thus, by Observation 3.1, we have that $\emptyset \neq e \cap \overline{V_1} \subseteq e \cap W$.

2. By definition, for every $i \in [2k-1]$, we have $u_{2k} \in H_{C-u_i} = \overline{A_i}$, and therefore $u_{2k} \in Z$. Thus, by Observation 3.1, we have that $e \cap Z \neq \emptyset$.

3. For every $i \in [2k-1]$, let $Y_i := A_i \setminus W$. We note that $(Y_1, \ldots, Y_{2k-1}, W, Z)$ is a partition of V. Therefore, in order to show that $e \subseteq W \cup Z$, it suffices to show that $e \cap Y_i = \emptyset$ for every $i \in [2k-1]$. By Observation 3.1, we know that $e \subseteq \overline{H_S}$. We will show that $\overline{H_S} \cap Y_i = \emptyset$ for every $i \in [2k-1]$ which implies that $e \cap Y_i = \emptyset$ for every $i \in [2k-1]$.

Let us fix an index $i \in [2k-1]$. We note that

$$Y_i = A_i \setminus W = A_i \setminus \left(\bigcup_{1 \le a < b \le 2k-1} (A_a \cap A_b) \right) = A_i \setminus \left(\bigcup_{1 \le a < b \le 2k-1} ((A_a \cap A_b) \cap A_i) \right) \tag{3.29}$$

$$= A_i \setminus \left(\bigcup_{j \in [2k-1] \setminus \{i\}} (A_j \cap A_i) \right) = A_i \setminus \left(\bigcup_{j \in [2k-1] \setminus \{i\}} A_j \right) = A_i \cap \overline{\left(\bigcup_{j \in [2k-1] \setminus \{i\}} A_j \right)} \tag{3.30}$$

$$= A_i \cap \left(\bigcap_{j \in [2k-1] \setminus \{i\}} \overline{A_j} \right) = \left(\bigcap_{j \in [2k-1] \setminus \{i\}} \overline{A_j} \right) \setminus \overline{A_i}. \tag{3.31}$$

Therefore,

$$Y_i \cap \overline{H_S} = \left(\left(\bigcap_{j \in [2k-1] \setminus \{i\}} \overline{A_j} \right) \setminus \overline{A_i} \right) \cap \overline{H_S} = \left(\left(\bigcap_{j \in [2k-1] \setminus \{i\}} \overline{A_j} \right) \setminus \overline{A_i} \right) \setminus H_S \tag{3.32}$$

$$= \left(\bigcap_{j \in [2k-1] \setminus \{i\}} H_{C-u_j} \right) \setminus \left(H_{C-u_i} \cup H_{C-u_{2k}} \right). \tag{3.33}$$

By Claim 3.6, we have that $H_{C-u_j} \subseteq H_{C-u_i} \cup H_{C-u_{2k}}$ for every $j \in [2k-1] \setminus \{i\}$. Therefore, $H_{C-u_j} \setminus (H_{C-u_i} \cup H_{C-u_{2k}}) = \emptyset$ for every $j \in [2k-1] \setminus \{i\}$, and hence,

$$\left(\bigcap_{j \in [2k-1] \setminus \{i\}} H_{C-u_j} \right) \setminus (H_{C-u_i} \cup H_{C-u_{2k}}) = \emptyset. \tag{3.34}$$

Thus, we have $Y_i \cap \overline{H_S} = \emptyset$.

QED.

By Claim 3.7, the hyperedge e satisfies the conditions of the second conclusion of Theorem 3.5. Therefore, by Theorem 3.5, there exists a k-partition $\mathcal{P}'$ with

$$\text{cost}(\mathcal{P}') < \frac{1}{2} \min\{d(A_i) + d(A_j) : i, j \in [2k-1], i \neq j\} \tag{3.35}$$

$$= d(V_1) \qquad \text{(By Claim 3.4)} \tag{3.36}$$

$$= \text{cost}(\mathcal{P}). \qquad \text{(By assumption of the theorem)} \tag{3.37}$$

Thus, we have obtained a k-partition $\mathcal{P}'$ with $\text{cost}(\mathcal{P}') < \text{cost}(\mathcal{P})$, which is a contradiction since $\mathcal{P}$ is a minimum k-partition. QED.

3.5 STRONGER STRUCTURAL THEOREM FOR $k = 2$

We prove the stronger version of Theorem 3.3 for $k = 2$—namely Theorem 3.4—in this section. We were able to prove Theorem 3.4 via two more techniques that are different from the one presented in this section—one technique is via a novel three-cut-set-lemma while the second technique is via the *canonical decomposition of hypergraphs* [38, 117, 123, 124]. It is unclear how to generalize both these techniques to $k \geq 3$. Here, we present a proof of Theorem 3.4 that closely resembles the proof of Theorem 3.3. We give an alternative proof of this theorem based on the three-cut-set-lemma in Section 3.6.

We will again use the containment lemma (Lemma 3.2) in our proof. We mention how we obtain the stronger statement relative to Theorem 3.3 in the proof below. We restate and prove Theorem 3.4 now.

Theorem 3.4. Let $G = (V, E)$ be a hypergraph and $\mathcal{P} = (V_1, V_2)$ be a minimum cut. Then, for all non-empty subsets $T \subseteq V_2$, there exists a subset $S \subseteq V_1$ with $|S| \leq 2$ such that the source minimal minimum (S, T)-terminal cut $(A, \overline{A})$ satisfies $\delta(A) = \delta(V_1)$ and $A \subseteq V_1$.

Proof. Let us fix an arbitrary non-empty subset $T \subseteq \overline{V_1} = V_2$. For a subset $X \subseteq V_1$, we denote the source minimal minimum (X, T)-terminal cut by $(H_X, \overline{H_X})$. By Lemma 3.2, for all $X \subseteq V_1$ we have that $H_X \subseteq V_1$. If $|V_1| \leq 2$, then choosing $S = V_1$ proves the theorem. So, we will assume henceforth that $|V_1| \geq 3$. We will show that there exists a subset $S \subseteq V_1$ with $|S| \leq 2$ such that the source minimal minimum (S, T)-terminal cut $(H_S, \overline{H_S})$ satisfies $\delta(H_S) = \delta(V_1)$. This suffices since we have that $H_S \subseteq V_1$ for all subsets $S \subseteq V_1$ (by Lemma 3.2).

We begin with the following useful claim. We note that Claim 3.8 crucially relies on the fact that V_1 is a part of a minimum cut (i.e, it crucially relies on $k = 2$)—it does not hold if V_1 is a part of a minimum k-partition for $k \geq 3$. Claim 3.8 serves a similar role in our proof to that of Claim 3.4 in the proof of Theorem 3.3. Unlike Claim 3.4, Claim 3.8 places no restriction on the size of the sets involved. This is important for obtaining the stronger structural result of Theorem 3.4.

Claim 3.8. For every $X \subseteq V_1$, we have that $d(H_X) = d(V_1)$.

Proof. Since $X \subseteq V_1$ and $T \subseteq V_2$, we have that (V_1, V_2) is a (X, T)-terminal cut. Since $(H_X, \overline{H_X})$ is a minimum (X, T)-terminal cut, we have that $d(H_X) \leq d(V_1)$. Since $(H_X, \overline{H_X})$

is a cut, and (V_1, V_2) is a minimum cut, we have that $d(H_X) \geq d(V_1)$. Thus, $d(H_X) = d(V_1)$. QED.

For the sake of contradiction, suppose that for every $S \subseteq V_1$ with $|S| \leq 2$, the source minimal minimum (S, T)-terminal cut $(H_S, \overline{H_S})$ does not satisfy $\delta(H_S) = \delta(V_1)$. Our proof strategy is to obtain a cheaper cut than (V_1, V_2), thereby contradicting the optimality of (V_1, V_2).

Let $S \subseteq V_1$ be a set of size 2 such that H_S is maximal—i.e., there does not exist $S' \subseteq V_1$ of size 2 such that $H_{S'} \supsetneq H_S$. In contrast to the proof of Theorem 3.3, where subsets S of size 3 had to be considered, here we only consider subsets S of size 2. We will see that this suffices to arrive at a contradiction. Let $S := \{u_1, u_2\}$. By assumption, we have that $\delta(H_S) \neq \delta(V_1)$, but by Claim 3.8, we have that $d(H_S) = d(V_1)$. Therefore, $\delta(V_1) \setminus \delta(H_S)$ is non-empty. Let $e \in \delta(V_1) \setminus \delta(H_S)$. Let $u_3 \in e \cap V_1$. Let $C := \{u_1, u_2, u_3\}$. For notational convenience we will use $C - u_i$ to denote $C \setminus \{u_i\}$ and $C - u_i - u_j$ to denote $C \setminus \{u_i, u_j\}$ for all $i, j \in [3]$. The choice of the hyperedge e is crucial to our proof—its properties will be used much later in our proof. We summarize the properties of the hyperedge e here.

Observation 3.2. The hyperedge e has the following properties:

1. $e \cap V_2 \neq \emptyset$
2. $u_3 \in e$, and
3. $e \subseteq \overline{H_S}$,

Our strategy to arrive at a cheaper cut than (V_1, V_2) is to apply the second conclusion of Theorem 3.5. The next few claims will set us up to obtain sets that satisfy the hypothesis of Theorem 3.5. We note that the following claim and its proof are identical to Claim 3.2 and its proof with $2k$ replaced by 3.

Claim 3.9. For every $i \in [3]$, we have $u_i \notin H_{C-u_i}$.

Proof. By Observation 3.2 we have $u_3 \in e$ and $e \subseteq \overline{H_S}$, so $u_3 \notin H_S = H_{C-u_3}$. Suppose $i \in [2]$. Our proof will rely on the choice of S.

Suppose for contradiction that $u_i \in H_{C-u_i}$ for some $i \in [2]$. Then we have that $S \subseteq H_{C-u_i}$, so $(H_{C-u_i} \cap H_S, \overline{H_{C-u_i} \cap H_S})$ is a (S, T)-terminal cut. Therefore,

$$d(H_{C-u_i} \cap H_S) \geq d(H_S). \tag{3.38}$$

Also, since $(H_{C-u_i} \cup H_S, \overline{H_{C-u_i} \cup H_S})$ is a $(C-u_i, T)$-terminal cut, we have that

$$d(H_{C-u_i} \cup H_S) \geq d(H_{C-u_i}). \tag{3.39}$$

By submodularity of the hypergraph cut function and inequalities (3.38) and (3.39), we have that

$$d(H_S) + d(H_{C-u_i}) \geq d(H_{C-u_i} \cap H_S) + d(H_{C-u_i} \cup H_S) \geq d(H_S) + d(H_{C-u_i}). \tag{3.40}$$

Therefore, inequality (3.38) is an equation, and consequently, $(H_{C-u_i} \cap H_S, \overline{H_{C-u_i} \cap H_S})$ is a minimum (S,T)-terminal cut. If $H_{C-u_i} \cap H_S \subsetneq H_S$, then this contradicts the source minimality of the minimum (S,T)-terminal cut $(H_S, \overline{H_S})$. Therefore, $H_{C-u_i} \cap H_S = H_S$ and hence, $H_S \subseteq H_{C-u_i}$. Also, the vertex u_3 is in $C-u_i$ but not in H_S and hence, $H_S \subsetneq H_{C-u_i}$. However, $|C-u_i| = 2$. Therefore, the set $C-u_i$ contradicts the choice of S. QED.

The next two claims will help in arguing properties about the hyperedge e which will allow us to use the second conclusion of Theorem 3.5. We note the similarity of Claims 3.10 and 3.11 in this proof to Claims 3.5 and 3.6 in the proof of Theorem 3.3. In order to prove Claims 3.5 and 3.6, we needed the size of C to be $2k$ in order to invoke Claim 3.4. Here, we are able to prove Claims 3.10 and 3.11 with the size of C being $2k-1$ (for $k=2$). This is because we can use Claim 3.8 (which holds only for $k=2$) instead of Claim 3.4, and Claim 3.8 does not require the size of C to be $2k$.

Claim 3.10. For every $i, j \in [3]$, we have

$$d(H_{C-u_i} \cap H_{C-u_j}) = d(V_1) = d(H_{C-u_i} \cup H_{C-u_j}).$$

Proof. Since $(H_{C-u_i} \cap H_{C-u_j}, \overline{H_{C-u_i} \cap H_{C-u_j}})$ is a $(C-u_i-u_j, T)$-terminal cut, we have that $d(H_{C-u_i} \cap H_{C-u_j}) \geq d(H_{C-u_i-u_j})$. By Claim 3.8, we have that $d(H_{C-u_i-u_j}) = d(V_1) = d(H_{C-u_i})$. Therefore,

$$d(H_{C-u_i} \cap H_{C-u_j}) \geq d(H_{C-u_i}). \tag{3.41}$$

Since $(H_{C-u_i} \cup H_{C-u_j}, \overline{H_{C-u_i} \cup H_{C-u_j}})$ is a $(C-u_j, T)$-terminal cut, we have that

$$d(H_{C-u_i} \cup H_{C-u_j}) \geq d(H_{C-u_j}). \tag{3.42}$$

By submodularity of the hypergraph cut function and inequalities (3.41) and (3.42), we have

that

$$d(H_{C-u_i})+d(H_{C-u_j}) \geq d(H_{C-u_i}\cap H_{C-u_j})+d(H_{C-u_i}\cup H_{C-u_j}) \geq d(H_{C-u_i})+d(H_{C-u_j}). \quad (3.43)$$

Therefore, inequalities (3.19) and (3.20) are equations. Thus, by Claim 3.8 we have that

$$d(H_{C-u_i} \cap H_{C-u_j}) = d(H_{C-u_i}) = d(V_1), \quad (3.44)$$

and

$$d(H_{C-u_i} \cup H_{C-u_j}) = d(H_{C-u_j}) = d(V_1). \quad (3.45)$$

QED.

The next claim follows from Claim 3.10 similar to the proof of Claim 3.6 from Claim 3.5 earlier. We include the proof for the sake of completeness.

Claim 3.11. For every $i, j, \ell \in [3]$ with $i \neq j$, we have $H_{C-u_\ell} \subseteq H_{C-u_i} \cup H_{C-u_j}$.

Proof. If $\ell = i$ or $\ell = j$ the claim is immediate. Thus, we assume that $\ell \neq i, j$. Let $Q := H_{C-u_\ell} \setminus (H_{C-u_i} \cup H_{C-u_j})$. We need to show that $Q = \emptyset$. We will show that $(H_{C-u_\ell} \setminus Q, \overline{H_{C-u_\ell} \setminus Q})$ is a minimum $(C - u_\ell, T)$-terminal cut. Consequently, Q must be empty (otherwise, $H_{C-u_\ell} \setminus Q \subsetneq H_{C-u_\ell}$ and hence, $(H_{C-u_\ell} \setminus Q, \overline{H_{C-u_\ell} \setminus Q})$ contradicts source minimality of the minimum $(C - u_\ell, T)$-terminal cut $(H_{C-u_\ell}, \overline{H_{C-u_\ell}})$).

We now show that $(H_{C-u_\ell} \setminus Q, \overline{H_{C-u_\ell} \setminus Q})$ is a minimum $(C - u_\ell, T)$-terminal cut. Since $H_{C-u_\ell} \setminus Q = H_{C-u_\ell} \cap (H_{C-u_i} \cup H_{C-u_j})$, we have that $C - u_i - u_j - u_\ell \subseteq H_{C-u_\ell} \setminus Q$. We also know that u_i and u_j are contained in both H_{C-u_ℓ} and $H_{C-u_i} \cup H_{C-u_j}$. Therefore, $C - u_\ell \subseteq H_{C-u_\ell} \setminus Q$. Thus, $(H_{C-u_\ell} \setminus Q, \overline{H_{C-u_\ell} \setminus Q})$ is a $(C - u_\ell, T)$-terminal cut. Therefore,

$$d(H_{C-u_\ell} \cap (H_{C-u_i} \cup H_{C-u_j})) = d(H_{C-u_\ell} \setminus Q) \geq d(H_{C-u_\ell}). \quad (3.46)$$

We also have that $(H_{C-u_\ell} \cup (H_{C-u_i} \cup H_{C-u_j}), \overline{H_{C-u_\ell} \cup (H_{C-u_i} \cup H_{C-u_j})})$ is a $(C - u_i, T)$-terminal cut. Therefore, $d(H_{C-u_\ell} \cup (H_{C-u_i} \cup H_{C-u_j})) \geq d(H_{C-u_i})$. By Claims 3.8 and 3.10, we have that $d(H_{C-u_i}) = d(V_1) = d(H_{C-u_i} \cup H_{C-u_j})$. Therefore,

$$d(H_{C-u_\ell} \cup (H_{C-u_i} \cup H_{C-u_j})) \geq d(H_{C-u_i} \cup H_{C-u_j}). \quad (3.47)$$

By submodularity of the hypergraph cut function and inequalities (3.46) and (3.47), we have

that

$$
\begin{aligned}
&d(H_{C-u_\ell}) + d(H_{C-u_i} \cup H_{C-u_j}) &(3.48)\\
&\geq d(H_{C-u_\ell} \cap (H_{C-u_i} \cup H_{C-u_j})) + d(H_{C-u_\ell} \cup (H_{C-u_i} \cup H_{C-u_j})) &(3.49)\\
&\geq d(H_{C-u_\ell}) + d(H_{C-u_i} \cup H_{C-u_j}). &(3.50)
\end{aligned}
$$

Therefore, inequalities (3.46) and (3.47) are equations, so $(H_{C-u_\ell} \setminus Q, \overline{H_{C-u_\ell} \setminus Q})$ is a minimum $(C - u_\ell, T)$-terminal cut. QED.

Let $R := \{u_3\}$, $U := V_1$, and for every $i \in [2]$, let $(\overline{A_i}, A_i) := (H_{C-u_i}, \overline{H_{C-u_i}})$. By Lemma 3.2, we have that $(\overline{A_i}, A_i)$ is a minimum $(C - u_i, V_2)$-terminal cut for every $i \in [2]$. Moreover, by Claim 3.9, we have that $u_i \in A_i \setminus A_{3-i}$. Hence, the sets U, R, and S, and the cuts $(\overline{A_i}, A_i)$ for $i \in [2]$ satisfy the conditions of Theorem 3.5. We will use the second conclusion of Theorem 3.5. We now show that the hyperedge e that we fixed at the beginning of the proof satisfies the conditions mentioned in the second conclusion of Theorem 3.5. We will use Claim 3.11 to prove this. Let $W := A_1 \cap A_2$ and $Z := \overline{A_1} \cap \overline{A_2}$ as in the statement of Theorem 3.5.

Claim 3.12. The hyperedge e satisfies the following conditions:

1. $e \cap W \neq \emptyset$,

2. $e \cap Z \neq \emptyset$, and

3. $e \subseteq W \cup Z$.

Proof. 1. By Lemma 3.2, for every $i \in [2]$ we have $V_2 \subseteq A_i$, and therefore $V_2 \subseteq W$. Thus, by Observation 3.2, we have that $\emptyset \neq e \cap V_2 \subseteq e \cap W$.

2. By definition, for every $i \in [2]$, we have $u_3 \in H_{C-u_i} = \overline{A_i}$, and therefore $u_3 \in Z$. Thus, by Observation 3.2, we have that $e \cap Z \neq \emptyset$.

3. For every $i \in [2]$, let $Y_i := A_i \setminus W$. We note that (Y_1, Y_2, W, Z) is a partition of V. Therefore, in order to show that $e \subseteq W \cup Z$, it suffices to show that $e \cap Y_1 = e \cap Y_2 = \emptyset$. By Observation 3.2, we know that $e \subseteq \overline{H_S}$. We will show that $\overline{H_S} \cap Y_i = \emptyset$ for every $i \in [2]$ which implies that $e \cap Y_i = \emptyset$ for every $i \in [2]$. Let us fix a $i \in [2]$. We note that

$$Y_i = A_i \setminus W = A_i \setminus (A_1 \cap A_2) = A_i \setminus A_{3-i} = A_i \cap \overline{A_{3-i}} = \overline{A_{3-i}} \setminus \overline{A_i}. \qquad (3.51)$$

Therefore,

$$Y_i \cap \overline{H_S} = \left(\overline{A_{3-i}} \setminus \overline{A_i}\right) \cap \overline{H_S} = \left(\overline{A_{3-i}} \setminus \overline{A_i}\right) \setminus H_S = H_{C-u_{3-i}} \setminus \left(H_{C-u_i} \cup H_{C-u_3}\right). \quad (3.52)$$

By Claim 3.11, we have that $H_{C-u_{3-i}} \subseteq H_{C-u_i} \cup H_{C-u_3}$. Therefore,

$$H_{C-u_{3-i}} \setminus (H_{C-u_i} \cup H_{C-u_3}) = \emptyset. \quad (3.53)$$

Thus, for every $i \in [2]$, we have $Y_i \cap \overline{H_S} = \emptyset$.

QED.

By Claim 3.12, the hyperedge e satisfies the conditions of the second conclusion of Theorem 3.5. Therefore, by Theorem 3.5, there exists a cut (V_1', V_2') with

$$\text{cost}(V_1', V_2') = d(V_1') < \frac{1}{2}(d(A_1) + d(A_2)) = d(V_1). \quad (3.54)$$

The last equality above is by Claim 3.8. Thus, we have obtained a cut (V_1', V_2') with $d(V_1') < d(V_1)$, which is a contradiction since (V_1, V_2) is a minimum cut. QED.

3.6 ALTERNATIVE PROOF OF STRONGER STRUCTURAL THEOREM FOR $k = 2$

In this section we give an alternative proof of Theorem 3.4. This alternative proof was the first proof that we discovered for Theorem 3.4, however we are unable to generalize its proof technique to $k \geq 2$ (i.e., to prove theorem 3.3). We present this alternative proof since we believe that it is based on a novel 3-cut-set lemma which may be of independent interest. Moreover, it is self-contained and does not rely on Theorem 3.5 (as opposed to the proof presented in Section 3.5).

Notation. Let $G = (V, E)$ be a hypergraph and let $R, S, T, U \subseteq V$ be subsets of vertices. We define

$$E[S] := \{e \in E \colon e \subseteq S\}, \quad (3.55)$$
$$E(S,T) := \{e \in E \colon e \subseteq S \cup T \text{ and } e \cap S, e \cap T \neq \emptyset\}, \quad (3.56)$$
$$E(S,T,U) := \{e \in E \colon e \subseteq S \cup T \cup U \text{ and } e \cap S, e \cap T, e \cap U \neq \emptyset\}, \text{ and} \quad (3.57)$$
$$E(R,S,T,U) := \{e \in E \colon e \subseteq R \cup S \cup T \cup U \text{ and } e \cap R, e \cap S, e \cap T, e \cap U \neq \emptyset\}. \quad (3.58)$$

Our proof approach is similar to the one used in [115] to prove an analogous structural theorem for graphs. Their proof relies on the *submodular triple inequality* which states that for every graph $G = (V, E)$ and every $X, Y, Z \subseteq V$, we have

$$d((Y \cap Z) \setminus X) + d((X \cap Z) \setminus Y) + d((X \cap Y) \setminus Z) + d(V \setminus (X \cup Y \cup Z)) \leq d(X) + d(Y) + d(Z). \quad (3.59)$$

Unfortunately, the submodular triple inequality fails to hold in hypergraphs (e.g., consider a hypergraph $G = (V, E)$ where $V = \{1, 2, 3, 4\}$, and $E = \{V\}$, with $X = \{1, 2\}$, $Y = \{1, 3\}$, and $Z = \{2, 3\}$). Our proof of Theorem 3.4 instead relies on the following novel Hypergraph 3-cut-set lemma.

Lemma 3.3 (Hypergraph 3-cut-set lemma)**.** Let $G = (V, E)$ be a hypergraph, and let $X, Y, Z \subseteq V$ be non-empty and pairwise disjoint sets such that $(X, \overline{X}), (Y, \overline{Y}), (Z, \overline{Z}), (X \cup Y, \overline{X \cup Y}), (X \cup Z, \overline{X \cup Z}), (Y \cup Z, \overline{Y \cup Z})$, and $(X \cup Y \cup Z, \overline{X \cup Y \cup Z})$ are all minimum cuts. Then,

$$\delta(X) = \delta(Y) = \delta(Z). \quad (3.60)$$

Proof. Let $W := \overline{X \cup Y \cup Z}$. Since $(X \cup Y \cup Z, \overline{X \cup Y \cup Z})$ is a minimum cut, $W \neq \emptyset$. Since $(X, \overline{X})$ and $(X \cup Y, \overline{X \cup Y})$ are both minimum cuts, we have that $d(X) = d(X \cup Y)$. The hyperedges which are in $\delta(X)$ but not $\delta(X \cup Y)$ are the hyperedges of $E(X, Y)$. The hyperedges which are in $\delta(X \cup Y)$ but not $\delta(X)$ are the hyperedges of $E(Y, Z)$, $E(Y, W)$ and $E(Y, Z, W)$. Thus, we have that

$$|E(X, Y)| = |E(Y, Z)| + |E(Y, W)| + |E(Y, Z, W)|. \quad (3.61)$$

We can symmetrically derive versions of Equation (3.61) for $|E(X, Z)|$, $|E(Y, X)|$, $|E(Y, Z)|$, $|E(Z, X)|$, and $|E(Z, Y)|$. Summing these six equations together we have that

$$\sum_{A,B \in \{X,Y,Z\}, A \neq B} |E(A, B)| = \left(\sum_{A,B \in \{X,Y,Z\}, A \neq B} |E(A, B)| \right) + \sum_{A,B \in \{X,Y,Z\}, A \neq B} \left(|E(A, W)| + |E(A, B, W)| \right). \quad (3.62)$$

Thus, we have that

$$\sum_{A,B \in \{X,Y,Z\}, A \neq B} \left(|E(A, W)| + |E(A, B, W)| \right) = 0. \quad (3.63)$$

Therefore we have that for every $A, B \in \{X, Y, Z\}$ with $A \neq B$, $E(A, W) = E(A, B, W) = \emptyset$.

Therefore, we have that

$$d(X) = |E(X,Y)| + |E(X,Z)| + |E(X,W)| + \tag{3.64}$$

$$|E(X,Y,Z)| + |E(X,Y,W)| + |E(X,Z,W)| + |E(X,Y,Z,W)| \tag{3.65}$$

$$= |E(X,Y)| + |E(X,Z)| + |E(X,Y,Z)| + |E(X,Y,Z,W)|. \tag{3.66}$$

We also have that

$$d(X \cup Y \cup Z) = |E(X,W)| + |E(Y,W)| + |E(Z,W)| + \tag{3.67}$$

$$|E(X,Y,W)| + |E(X,Z,W)| + |E(Y,Z,W)| + |E(X,Y,Z,W)| \tag{3.68}$$

$$= |E(X,Y,Z,W)|. \tag{3.69}$$

Since $(X \cup Y \cup Z, \overline{X \cup Y \cup Z})$ is a minimum cut, we have that $d(X) = d(X \cup Y \cup Z)$. Thus, by the two equations above, we have that

$$|E(X,Y)| + |E(X,Z)| + |E(X,Y,Z)| + |E(X,Y,Z,W)| = |E(X,Y,Z,W)|. \tag{3.70}$$

Thus,

$$|E(X,Y)| + |E(X,Z)| + |E(X,Y,Z)| = 0. \tag{3.71}$$

Therefore we have that $E(X,Y) = E(X,Z) = E(X,Y,Z) = \emptyset$. By a symmetric argument, we can conclude that $E(Y,Z) = \emptyset$. Then, we have that

$$\delta(X) = \delta(Y) = \delta(Z) = \delta(X \cup Y \cup Z) = E(X,Y,Z,W). \tag{3.72}$$

QED.

We will repeatedly use the well-known uncrossing result given in Lemma 3.4 below. We present a proof for the sake of completeness. We emphasize that the lemma relies only on the submodularity of the hypergraph cut function and in fact, holds even for submodular functions. We say that two sets A, B cross if $A \cap B$, $A \cap \overline{B}$, $\overline{A} \cap B$, and $\overline{A} \cap \overline{B}$ are all non-empty.

Lemma 3.4. Let $G = (V,E)$ be a hypergraph. Let $A, B \subseteq V$ and $\emptyset \subsetneq S \subseteq A \cap B$, $\emptyset \subsetneq T \subseteq \overline{A} \cap \overline{B}$ be sets such that $(A, \overline{A})$ and $(B, \overline{B})$ are minimum (S,T)-terminal cuts. If A and B cross, then $(A \cap B, \overline{A \cap B})$ and $(A \cup B, \overline{A \cup B})$ are also minimum (S,T)-terminal cuts.

Proof. Since $S \subseteq A \cap B$ and $T \subseteq \overline{A} \cap \overline{B}$, we have that $(A \cap B, \overline{A \cap B})$ and $(A \cup B, \overline{A \cup B})$ are

(S,T)-terminal cuts. Since $(A,\overline{A})$ and $(B,\overline{B})$ are minimum (S,T)-terminal cuts, we have that

$$d(A) + d(B) \leq d(A \cap B) + d(A \cup B). \tag{3.73}$$

Since the cut function of a hypergraph is submodular, we also have that

$$d(A \cap B) + d(A \cup B) \leq d(A) + d(B). \tag{3.74}$$

Thus, we conclude that

$$d(A \cap B) + d(A \cup B) = d(A) + d(B), \tag{3.75}$$

and therefore $(A \cap B, \overline{A \cap B})$ and $(A \cup B, \overline{A \cup B})$ are both minimum (S,T)-terminal cuts. QED.

Corollary 3.3. Let $A, B \subseteq V$ be sets such that $(A,\overline{A})$ and $(B,\overline{B})$ are minimum cuts. If A and B cross, then $(A \cap B, \overline{A \cap B})$ and $(A \cup B, \overline{A \cup B})$ are also minimum cuts.

We now restate and prove Theorem 3.4.

Theorem 3.4. Let $G = (V, E)$ be a hypergraph and $\mathcal{P} = (V_1, V_2)$ be a minimum cut. Then, for all non-empty subsets $T \subseteq V_2$, there exists a subset $S \subseteq V_1$ with $|S| \leq 2$ such that the source minimal minimum (S,T)-terminal cut $(A,\overline{A})$ satisfies $\delta(A) = \delta(V_1)$ and $A \subseteq V_1$.

Proof. Let $T \subseteq V$ be arbitrary and let $\mathcal{H} := \{U \subseteq V \setminus T : (U,\overline{U}) \text{ is a minimum cut in } G\}$. Suppose for contradiction that the theorem does not hold for some $V_1 \in \mathcal{H}$. For every $a, b \in V_1$, let H_{ab} be the minimal set in $\mathcal{H}$ such that $a, b \in H_{ab}$ and $\delta(H_{ab}) \neq \delta(V_1)$ (a set H_{ab} with $a, b \in H_{ab}$ and $\delta(H_{ab}) \neq \delta(V_1)$ exists since we have assumed that the theorem does not hold for V_1).

Claim 3.13. $H_{ab} \subseteq V_1$ for every $a, b \in V_1$.

Proof. Suppose for contradiction that there exist $a, b \in V_1$ such that $H_{ab} \not\subseteq V_1$. Then $H_{ab} \cap \overline{V_1} \neq \emptyset$. If $\overline{H_{ab}} \cap V_1 = \emptyset$, then $V_1 \subseteq H_{ab}$, and hence, V_1 contradicts the minimality of H_{ab}. So, $\overline{H_{ab}} \cap V_1 \neq \emptyset$. Since $a \in H_{ab} \cap V_1$, and $T \subseteq \overline{H_{ab}} \cap \overline{V_1}$, the sets H_{ab} and V_1 cross. Consequently, by Corollary 3.3, the partition $(H_{ab} \cap V_1, \overline{H_{ab} \cap V_1})$ is also a minimum cut, and thus $H_{ab} \cap V_1 \in \mathcal{H}$. This contradicts the minimal choice of H_{ab}. QED.

Claim 3.14. There exist distinct vertices $x, y, z \in V_1$ such that $x \notin H_{yz}$, $y \notin H_{xz}$, $z \notin H_{xy}$, and the three hyperedge sets $\delta(H_{xy} \cap H_{xz})$, $\delta(H_{xy} \cap H_{yz})$, and $\delta(H_{xz} \cap H_{yz})$ are not all equal.

Proof. Let $a, b \in V_1$ be vertices such that $|H_{ab}|$ is maximum. By the definition of H_{ab}, we have that $\delta(H_{ab}) \neq \delta(V_1)$, and since $(H_{ab}, \overline{H_{ab}})$ is a minimum cut we must have $d(H_{ab}) = d(V_1)$. Therefore, $\delta(V_1) \setminus \delta(H_{ab}) \neq \emptyset$, so let $e \in \delta(V_1) \setminus \delta(H_{ab})$, and let $z \in e \cap V_1$. By Claim 3.13, we have that $H_{ab} \subseteq V_1$. Since $z \in e$ and $e \in \delta(V_1) \setminus \delta(H_{ab})$, we have that $z \notin H_{ab}$. Let $p \in H_{ab}$ be a vertex such that $|H_{pz} \cap H_{ab}|$ is maximum. Since $z \in H_{pz} \setminus H_{ab}$, we have that $H_{ab} \neq H_{pz}$. Thus, by the choice of a and b, we have that $H_{ab} \not\subseteq H_{pz}$. Therefore, $H_{ab} \setminus H_{pz} \neq \emptyset$. Let $y \in H_{ab} \setminus H_{pz}$. By our choice of p, we have that $|H_{pz} \cap H_{ab}| \geq |H_{yz} \cap H_{ab}|$. Since $y \in (H_{yz} \cap H_{ab}) \setminus (H_{pz} \cap H_{ab})$, we have that $(H_{pz} \cap H_{ab}) \setminus H_{yz} \neq \emptyset$. Let $x \in (H_{pz} \cap H_{ab}) \setminus H_{yz}$.

We now analyze relationships between some of the sets that we have defined.

Proposition 3.2. $y \notin H_{xz}$.

Proof. Suppose for contradiction that $y \in H_{xz}$. Then, since $y \notin H_{pz}$, we have that $H_{xz} \cap \overline{H_{pz}} \neq \emptyset$. By the choice of p, we have that $|H_{pz} \cap H_{ab}| \geq |H_{xz} \cap H_{ab}|$. Since $y \in (H_{xz} \cap H_{ab}) \setminus H_{pz}$, we have that $(H_{pz} \cap H_{ab}) \setminus H_{xz} \neq \emptyset$. Thus, $\overline{H_{xz}} \cap H_{pz} \neq \emptyset$. Since $z \in H_{xz} \cap H_{pz}$, we have $H_{xz} \cap H_{pz} \neq \emptyset$. By Claim 3.13, we have $\overline{H_{xz}} \cap \overline{H_{pz}} \neq \emptyset$. Thus, H_{xz} and H_{pz} cross. Therefore, by Corollary 3.3, we have $H_{xz} \cap H_{pz} \in \mathcal{H}$. Since $H_{xz} \cap H_{pz} \subsetneq H_{xz}$ and $x, z \in H_{xz} \cap H_{pz}$, by the choice of x and z, the set $H_{xz} \cap H_{pz}$ contradicts the minimality of H_{xz}. Therefore, we conclude that $y \notin H_{xz}$. QED.

Proposition 3.3. $H_{xy} \subseteq H_{ab}$.

Proof. Suppose for contradiction that $H_{xy} \not\subseteq H_{ab}$. By our choice of a, b, we cannot have $H_{ab} \subseteq H_{xy}$ either. We know that $x, y \in H_{xy} \cap H_{ab}$, and since $H_{xy}, H_{ab} \subseteq V_1$ (by Claim 3.13), we have that $\overline{H_{xy}} \cap \overline{H_{ab}} \neq \emptyset$. Therefore, H_{xy} and H_{ab} cross. Thus, by Corollary 3.3, we have that $H_{ab} \cap H_{xy} \in \mathcal{H}$. Since $H_{xy} \cap H_{ab} \subsetneq H_{xy}$ and $x, y \in H_{ab} \cap H_{xy}$, we have that the set $H_{ab} \cap H_{xy}$ contradicts the minimality of H_{xy}. Thus, we conclude that $H_{xy} \subseteq H_{ab}$. QED.

We now show that $x \notin H_{yz}$, $y \notin H_{xz}$, and $z \notin H_{xy}$. We have the following facts:

1. By our choice of x, we have that $x \notin H_{yz}$.

2. By Proposition 3.2, we have that $y \notin H_{xz}$.

3. By our choice of z, we have that $z \notin H_{ab}$, and therefore by Proposition 3.3, we have that $z \notin H_{xy}$.

Finally we note that the hyperedge e which we selected when choosing z contains both z and some vertex r in $\overline{V_1}$. By definition, we have that $z \in H_{xz}, H_{yz}$. By Claim 3.13, $H_{xz}, H_{yz} \subseteq V_1$. Therefore, $e \in \delta(H_{xz} \cap H_{yz})$. By Claim 3.13, we have that $H_{ab} \subseteq V_1$, and hence, $r \notin H_{ab}$. However, by choice of e, we have that $e \notin \delta(H_{ab})$, and hence $e \cap H_{ab} = \emptyset$. Therefore, by Proposition 3.3, we have that $e \cap H_{xy} = \emptyset$, and hence $e \cap (H_{xy} \cap H_{xz}) = \emptyset$. Thus, $e \notin \delta(H_{xy} \cap H_{xz})$. Therefore, we conclude that $\delta(H_{xz} \cap H_{yz}) \neq \delta(H_{xy} \cap H_{xz})$, and hence the sets $\delta(H_{xy} \cap H_{xz}), \delta(H_{xy} \cap H_{yz})$, and $\delta(H_{xz} \cap H_{yz})$ are not all equal. This completes the proof of Claim 3.14. QED.

Claim 3.15. Let $x, y, z \in V_1$ be such that $x \notin H_{yz}$, $y \notin H_{xz}$, and $z \notin H_{xy}$. Then

$$H_{xy} \setminus (H_{xz} \cup H_{yz}) = H_{xz} \setminus (H_{xy} \cup H_{yz}) = H_{yz} \setminus (H_{xy} \cup H_{xz}) = \emptyset.$$

Proof. We will prove that $H_{xy} \setminus (H_{xz} \cup H_{yz}) = \emptyset$. The arguments for $H_{xz} \setminus (H_{xy} \cup H_{yz})$ and $H_{yz} \setminus (H_{xy} \cup H_{xz})$ are similar.

Let $Q := H_{xy} \setminus (H_{xz} \cup H_{yz})$. We note that $x \in H_{xy} \cap H_{xz}$, and therefore $H_{xy} \setminus Q$ is non-empty. Therefore $(H_{xy} \setminus Q, \overline{H_{xy} \setminus Q})$ is a cut. Since $(H_{xy}, \overline{H_{xy}})$ is a minimum cut, this means that

$$d(H_{xy}) \leq d(H_{xy} \setminus Q). \tag{3.76}$$

The hyperedges which are in $\delta(H_{xy})$ but not in $\delta(H_{xy} \setminus Q)$ are the hyperedges in $E(Q, \overline{H_{xy}})$. The hyperedges which are in $\delta(H_{xy} \setminus Q)$ but not $\delta(H_{xy})$ are the hyperedges in $E(Q, H_{xy} \setminus Q)$. Thus, inequality (3.76) implies that

$$|E(Q, \overline{H_{xy}})| \leq |E(Q, H_{xy} \setminus Q)|. \tag{3.77}$$

Next we note that $x \in H_{xz} \setminus H_{yz}$, $y \in H_{yz} \setminus H_{xz}$, $z \in H_{xz} \cap H_{yz}$, and, by Claim 3.13, $\overline{V_1} \subseteq \overline{H_{xz}} \cap \overline{H_{yz}}$. Therefore, H_{xz} and H_{yz} cross. Thus, by Corollary 3.3, we have that $H_{xz} \cup H_{yz} \in \mathcal{H}$. If $Q = \emptyset$ we are done. Otherwise, we have that $H_{xy} \cap \overline{(H_{xz} \cup H_{yz})} = Q \neq \emptyset$, $y \in H_{xy} \cap (H_{xz} \cup H_{yz})$, $z \in \overline{H_{xy}} \cap (H_{xz} \cup H_{yz})$, and $\overline{V_1} \subseteq \overline{H_{xy}} \cap \overline{(H_{xz} \cup H_{yz})}$. Thus, H_{xy} and $H_{xz} \cup H_{yz}$ cross. Therefore, by Corollary 3.3, we have that $H_{xy} \cup H_{xz} \cup H_{yz} \in \mathcal{H}$. Therefore, we have that

$$d(H_{xz} \cup H_{yz}) = d(H_{xy} \cup H_{xz} \cup H_{yz}) = d(Q \cup (H_{xz} \cup H_{yz})). \tag{3.78}$$

Since Q and $H_{xz} \cup H_{yz}$ are disjoint, we have that the hyperedges which are in $\delta(H_{xz} \cup H_{yz})$ but not in $\delta(Q \cup (H_{xz} \cup H_{yz}))$ are the hyperedges of $E(Q, H_{xz} \cup H_{yz})$, and the hyperedges which are in $\delta(Q \cup (H_{xz} \cup H_{yz}))$, but not $\delta(H_{xz} \cup H_{yz})$ are the hyperedges of $E(Q, \overline{H_{xy} \cup H_{xz} \cup H_{yz}})$. Thus, Equation (3.78) implies that

$$|E(Q, H_{xz} \cup H_{yz})| = |E(Q, \overline{H_{xy} \cup H_{xz} \cup H_{yz}})|. \tag{3.79}$$

Therefore, we have that

$$\begin{aligned}|E(Q, \overline{H_{xy}})| &\geq |E(Q, \overline{H_{xy} \cup H_{xz} \cup H_{yz}})| && (3.80)\\ &= |E(Q, H_{xz} \cup H_{yz})| && (3.81)\\ &\geq |E(Q, (H_{xz} \cup H_{yz}) \cap H_{xy})| && (3.82)\\ &= |E(Q, H_{xy} \setminus Q)| && (3.83)\\ &\geq |E(Q, \overline{H_{xy}})|. && (3.84)\end{aligned}$$

Here the equation on the second line comes from Equation (3.79) and the inequality on the last line comes from Inequality (3.77). Since the first and last terms in the inequality chain are the same, equality holds throughout. Therefore, $|E(Q, \overline{H_{xy}})| = |E(Q, H_{xy} \setminus Q)|$, and so $d(H_{xy}) = d(H_{xy} \setminus Q)$. Thus, $H_{xy} \setminus Q \in \mathcal{H}$. Since $x, y \in H_{xz} \cup H_{yz}$, $x, y \in H_{xy} \setminus Q$. Furthermore $H_{xy} \setminus Q \subseteq H_{xy}$. Since H_{xy} is a minimial element of $\mathcal{H}$ containing x and y, it must be that $H_{xy} \setminus Q = H_{xy}$, and thus $Q = \emptyset$. QED.

The next claim completes the proof of Theorem 3.4, because the conclusion of this claim is in direct contradiction with the conclusion of Lemma 3.3. QED.

Claim 3.16. There exist non-empty and pairwise disjoint sets $X, Y, Z \subseteq V_1$ such that $X, Y, Z, X \cup Y, X \cup Z, Y \cup Z, X \cup Y \cup Z \in \mathcal{H}$, and $\delta(X), \delta(Y), \delta(Z)$ are not all equal.

Proof. Let $x, y, z \in V_1$ be such that $x \notin H_{yz}$, $y \notin H_{xz}$, and $z \notin H_{xy}$, and the three hyperedge sets $\delta(H_{xy} \cap H_{xz}), \delta(H_{xy} \cap H_{yz})$, and $\delta(H_{xz} \cap H_{yz})$ are not all equal. Such x, y, z exist by Claim 3.14. Let

$$\begin{aligned}X' &:= H_{xy} \cap H_{xz}, && (3.85)\\ Y' &:= H_{xy} \cap H_{yz}, \text{ and} && (3.86)\\ Z' &:= H_{xz} \cap H_{yz}. && (3.87)\end{aligned}$$

By Claim 3.15, we have that $H_{xy} \setminus (H_{xz} \cup H_{yz})$, $H_{xz} \setminus (H_{xy} \cup H_{yz})$, and $H_{yz} \setminus (H_{xy} \cup H_{xz})$ are all empty, and hence $H_{xy} = X' \cup Y'$, $H_{xz} = X' \cup Z'$, and $H_{yz} = Y' \cup Z'$. Let

$$\begin{aligned}U &:= H_{xy} \cap H_{xz} \cap H_{yz}, && (3.88)\\ W &:= \overline{H_{xy} \cup H_{xz} \cup H_{yz}}, && (3.89)\\ X &:= X' \setminus U, && (3.90)\\ Y &:= Y' \setminus U, \text{ and} && (3.91)\\ Z &:= Z' \setminus U. && (3.92)\end{aligned}$$

Since H_{xy} and H_{xz} cross (because $x \in H_{xy} \cap H_{xz}$, $y \in H_{xy} \setminus H_{xz}$, and $z \in H_{xz} \setminus H_{xy}$), we have that $X' \in \mathcal{H}$. Since $x \notin H_{yz}$, we have that $x \notin U$, and hence $x \in X$, so $(X, \overline{X})$ is a cut. Therefore,

$$d(X) \geq d(X'). \tag{3.93}$$

The hyperedges which are in $\delta(X)$ but not $\delta(X')$ are the hyperedges of $E(X, U)$. The hyperedges which are in $\delta(X')$ but not $\delta(X)$ are the hyperedges of $E(U, Y)$, $E(U, Z)$, $E(U, W)$, $E(U, Y, Z)$, $E(U, Y, W)$, $E(U, Z, W)$, and $E(U, Y, Z, W)$. Thus, Equation (3.93) implies that

$$\begin{aligned} |E(X,U)| \geq {} & |E(U,Y)| + |E(U,Z)| + |E(U,W)| + \\ & |E(U,Y,Z)| + |E(U,Y,W)| + |E(U,Z,W)| + |E(U,Y,Z,W)|. \end{aligned} \tag{3.94}$$

We can derive symmetric inequalities for $|E(Y, U)|$ and $|E(Z, U)|$ by exchanging variable names in the preceding argument. Summing these three inequalities together, we have that

$$\begin{aligned} \sum_{A \in \{X,Y,Z\}} |E(A,U)| \geq 2 \sum_{A \in \{X,Y,Z\}} (|E(A,U)| + |E(A,U,W)|) + 3|E(U,W)| + \\ |E(U,X,Y)| + |E(U,X,Z)| + |E(U,Y,Z)| + \\ |E(U,X,Y,W)| + |E(U,X,Z,W)| + |E(U,Y,Z,W)|. \end{aligned} \tag{3.95}$$

Subtracting $|E(X, U)| + |E(Y, U)| + |E(Z, U)|$ from both sides of this inequality, we have that the sets $E(U, X), E(U, Y)$, $E(U, Z)$, $E(U, W)$, $E(U, Y, Z)$, $E(U, Y, W)$, $E(U, Z, W)$, and $E(U, Y, Z, W)$ are all empty. Thus, $\delta(X) = \delta(X')$, so $X \in \mathcal{H}$. Similarly, all of the hyperedges in $\delta(X' \cup Y')$ which are not in $\delta(X \cup Y)$ are from sets that we have concluded are empty, so $\delta(X \cup Y) = \delta(X' \cup Y') = \delta(H_{xy})$. Thus, $X \cup Y \in \mathcal{H}$. Symmetrically, $Y, Z, X \cup Z$, and $Y \cup Z$ are all in $\mathcal{H}$ as well. Since $X \cup Y$ and $Y \cup Z$ cross, we have that $X \cup Y \cup Z \in \mathcal{H}$, by Corollary 3.3. Since $x \in X$, $y \in Y$, $z \in Z$, X, Y, and Z are all non-empty. By definition, X, Y, and Z are disjoint. Since $\{\delta(X), \delta(Y), \delta(Z)\} = \{\delta(X'), \delta(Y'), \delta(Z')\} = \{\delta(H_{xy} \cap H_{xz}), \delta(H_{xy} \cap H_{yz}), \delta(H_{xz} \cap H_{yz})\}$, we have that the three hyperedge sets $\delta(X), \delta(Y), \delta(Z)$ are not all equal, by the guarantee that we obtained from Claim 3.14. Thus, X, Y, Z satisfy the conditions of the claim we are proving. QED.

3.7 PROOF OF THEOREM 3.5

We prove Theorem 3.5 in this section. We will need certain partition uncrossing and partition aggregation results from [73] that rely on more careful counting of hyperedges than simply employing the submodularity inequality. We begin with some notation that will help in such careful counting—our notation will be identical to the notation in [73]. Let $(Y_1, \dots, Y_p, W, Z)$ be a partition of V. We recall that $\text{cost}(Y_1, \dots, Y_p, W, Z)$ denotes the number of hyperedges that cross the partition. We define the following quantities:

1. Let $\text{cost}(W, Z) := |\{e \mid e \subseteq W \cup Z, e \cap W \neq \emptyset, e \cap Z \neq \emptyset\}|$ be the number of hyperedges contained in $W \cup Z$ that intersect both W and Z.

2. Let $\alpha(Y_1, \dots, Y_p, W, Z)$ be the number of hyperedges that intersect Z and at least two of the sets in $\{Y_1, \dots, Y_p, W\}$.

3. Let $\beta(Y_1, \dots, Y_p, Z)$ be the number of hyperedges that are disjoint from Z but intersect at least two of the sets in $\{Y_1, \dots, Y_p\}$.

For a partition $(Y_1, \dots, Y_p, W, Z)$, we will be interested in the sum of $\text{cost}(Y_1, \dots, Y_p, W, Z)$ with the three quantities defined above which we denote as $\sigma(Y_1, \dots, Y_p, W, Z)$, i.e.,

$$\begin{aligned}\sigma(Y_1, \dots, Y_p, W, Z) :=& \text{cost}(Y_1, \dots, Y_p, W, Z) + \text{cost}(W, Z) \\ &+ \alpha(Y_1, \dots, Y_p, W, Z) + \beta(Y_1, \dots, Y_p, Z). \qquad (3.96)\end{aligned}$$

The precise interpretation of the quantity $\sigma(Y_1, \dots, Y_p, W, Z)$ will not be important for our purposes—see [73] for the interpretation.

The following result from [73] shows that a collection of sets can be uncrossed to obtain a partition with small σ-value. We note the similarity of the hypothesis of Lemma 3.5 with the hypothesis of Theorem 3.5 and once again, refer to Figure 3.1 for an illustration of the sets that appear in the statement of Lemma 3.5.

Lemma 3.5. [73]. Let $G = (V, E)$ be a hypergraph and $\emptyset \neq R \subsetneq U \subsetneq V$. Let $S = \{u_1, \dots, u_p\} \subseteq U \setminus R$ for $p \geq 2$. Let $(\overline{A_i}, A_i)$ be a minimum $((S \cup R) \setminus \{u_i\}, \overline{U})$-terminal cut. Suppose that $u_i \in A_i \setminus (\cup_{j \in [p] \setminus \{i\}} A_j)$ for every $i \in [p]$. Let

$$Z := \cap_{i=1}^{p} \overline{A_i}, \; W := \cup_{1 \leq i < j \leq p} (A_i \cap A_j), \text{ and } Y_i := A_i - W \; \forall i \in [p].$$

Then, $(Y_1, \dots, Y_p, W, Z)$ is a $(p+2)$-partition of V with

$$\sigma(Y_1, \dots, Y_p, W, Z) \leq \min\{d(A_i) + d(A_j) : i, j \in [p], i \neq j\}.$$

Moreover, if $p = 2$, then the above inequality is an equation.

The next lemma from [73] will help in aggregating the parts of a ℓ-partition $\mathcal{P}$ where $\ell \geq 2k$ to a k-partition $\mathcal{K}$ while controlling the cost of $\mathcal{K}$.

Lemma 3.6. [73]. Let $G = (V, E)$ be a hypergraph, let $k \geq 2$ be an integer, and let $(Y_1, \ldots, Y_p, W, Z)$ be a partition of V for some integer $p \geq 2k-2$. Then, there exist distinct $i_1, \ldots, i_{k-1} \in [p]$ such that

$$\begin{aligned} 2\text{cost}\left(Y_{i_1}, \ldots, Y_{i_{k-1}}, V \setminus (\cup_{j=1}^{k-1} Y_{i_j})\right) \leq & \text{cost}(Y_1, \ldots, Y_p, W, Z) + \alpha(Y_1, \ldots, Y_p, W, Z) \\ & + \beta(Y_1, \ldots, Y_p, Z). \end{aligned}$$

We now restate and prove Theorem 3.5.

Theorem 3.5. Let $G = (V, E)$ be a hypergraph, $k \geq 2$ be an integer and $\emptyset \neq R \subsetneq U \subsetneq V$. Let $S = \{u_1, \ldots, u_p\} \subseteq U \setminus R$ for $p \geq 2k-2$. Let $(\overline{A_i}, A_i)$ be a minimum $((S \cup R) \setminus \{u_i\}, \overline{U})$-terminal cut. Suppose that $u_i \in A_i \setminus (\cup_{j \in [p] \setminus \{i\}} A_j)$ for every $i \in [p]$. Then, the following two hold:

1. There exists a k-partition $(P_1, \ldots, P_k)$ of V with $\overline{U} \subsetneq P_k$ such that
$$\text{cost}(P_1, \ldots, P_k) \leq \frac{1}{2} \min\{d(A_i) + d(A_j) : i, j \in [p], i \neq j\}.$$

2. Moreover, if there exists a hyperedge $e \in E$ such that e intersects $W := \cup_{1 \leq i < j \leq p}(A_i \cap A_j)$, e intersects $Z := \cap_{i \in [p]} \overline{A_i}$, and e is contained in $W \cup Z$, then the inequality in the previous conclusion is strict.

Proof. For the first conclusion of the theorem, we will use the same proof that appeared in [73]. We need the details of this proof to prove the second conclusion of the theorem.

We begin by proving the first conclusion. By applying Lemma 3.5, we obtain a $(p+2)$-partition $(Y_1, \ldots, Y_p, W, Z)$ such that

$$\sigma(Y_1, \ldots, Y_p, W, Z) \leq \min\{d(A_i) + d(A_j) : i, j \in [p], i \neq j\} \tag{3.97}$$

and moreover, $\overline{U} \subseteq W$, where $Y_i = A_i - W$ for all $i \in [p]$, $W = \cup_{1 \leq i < j \leq p}(A_i \cap A_j)$, and $Z = \cap_{i \in [p]} \overline{A_i}$. We recall that $p \geq 2k-2$. Hence, by applying Lemma 3.6 to the $(p+2)$-partition $(Y_1, \ldots, Y_p, W, Z)$, we obtain a k-partition $(P_1, \ldots, P_k)$ of V such that $W \cup Z \subseteq P_k$

and

$$
\begin{aligned}
&\operatorname{cost}(P_1, \ldots, P_k) \\
&\leq \frac{1}{2}\left(\operatorname{cost}(Y_1, \ldots, Y_p, W, Z) + \alpha(Y_1, \ldots, Y_p, W, Z) + \beta(Y_1, \ldots, Y_p, Z)\right) && (3.98) \\
&\leq \frac{1}{2}\left(\operatorname{cost}(Y_1, \ldots, Y_p, W, Z) + \operatorname{cost}(W, Z) + \alpha(Y_1, \ldots, Y_p, W, Z) + \beta(Y_1, \ldots, Y_p, Z)\right) && (3.99) \\
&= \frac{1}{2}\sigma(Y_1, \ldots, Y_p, W, Z) && (3.100) \\
&\leq \frac{1}{2}\min\{d(A_i) + d(A_j) : i, j \in [p], i \neq j\}. && (3.101)
\end{aligned}
$$

We note that $\overline{U}$ is strictly contained in P_k since $\overline{U} \cup Z \subseteq W \cup Z \subseteq P_k$ and Z is non-empty.

We now prove the second conclusion of the theorem. If there exists a hyperedge $e \in E$ such that e intersects W, e intersects Z, and e is contained in $W \cup Z$, then $\operatorname{cost}(W, Z) > 0$. Consequently, inequality (3.99) in the above sequence of inequalities should be strict. QED.

3.8 CONCLUSION AND OPEN PROBLEMS

Several works in the literature have approached global cut and partitioning problems via minimum (S, T)-terminal cuts (e.g., see [72, 73, 115, 133, 134]). Our work adds to this rich literature by showing that Enum-Hypergraph-k-Cut can be solved via minimum (S, T)-terminal cuts. In addition to the first deterministic polynomial-time algorithm for solving Enum-Hypergraph-k-Cut, one of the main contributions of this chapter is an elegant explanation for the number of minimum cut-sets in an n-vertex hypergraph being at most $\binom{n}{2}$ and a simple approach to enumerate all of them in a given hypergraph in deterministic polynomial time based on min (S, T)-cuts.

We note that our deterministic run-time to solve Enum-Hypergraph-k-Cut is $n^{O(k^2)}p$, where p is the size of the input hypergraph. Subsequent to the present work, we improved the deterministic run-time to $n^{O(k)}p$ in a separate paper [75]. In that paper, we strengthened Theorem 3.3 in a way that allowed it to solve Enum-Hypergraph-k-Cut as well as to enumerate all minmax k-cut-sets (a k-cut-set $F \subseteq E$ of a hypergraph $H = (V, E)$ is a minmax k-cut-set if there exists a partition $V_1, V_2, \ldots, V_k$ of V such that $\max_{i\in[k]} |\delta(V_i)| \leq \max_{i\in[k]} |\delta(P_i)|$ for every partition $P_1, P_2, \ldots, P_k$ of V and $F = \delta(V_1, \ldots, V_k)$).

We mention an open question raised by our work. For a long time, the known upper bound on the number of minimum k-partitions in connected graphs was $O(n^{2k-2})$ [65, 71, 106] while the known lower bound was $\Omega(n^k)$ (cycle), where n is the number of vertices in the input graph. A recent result improved the upper bound to $O(n^k)$ for fixed k which also resulted in a faster randomized algorithm to solve GRAPH-k-CUT for fixed k [66, 67]. We currently know that the number of minimum k-cut-sets in hypergraphs is $O(n^{2k-2})$ and is $\Omega(n^k)$ (the lower bound comes from graphs). Can we improve the upper bound on the number of minimum k-cut-sets in hypergraphs to $O(n^k)$?

CHAPTER 4: MULTICRITERIA MIN-CUT

In this chapter we address counting and optimization variants of multicriteria global min-cut and size-constrained min-k-cut in hypergraphs.

1. For an r-rank n-vertex hypergraph endowed with t hyperedge-cost functions, we show that the number of multiobjective min-cuts is $O(r2^{tr}n^{3t-1})$. In particular, this shows that the number of parametric min-cuts in constant rank hypergraphs for a constant number of criteria is strongly polynomial, thus resolving an open question by Aissi, Mahjoub, McCormick, and Queyranne [81]. In addition, we give randomized algorithms to enumerate all multiobjective min-cuts and all pareto-optimal cuts in strongly polynomial-time.

2. We also address node-budgeted multiobjective min-cuts: For an n-vertex hypergraph endowed with t vertex-weight functions, we show that the number of node-budgeted multiobjective min-cuts is $O(r2^{r}n^{t+2})$, where r is the rank of the hypergraph, and the number of node-budgeted b-multiobjective min-cuts for a fixed budget-vector $b \in \mathbb{R}^t_{\geq 0}$ is $O(n^2)$.

3. We show that min-k-cut in hypergraphs subject to constant lower bounds on part sizes is solvable in polynomial-time for constant k, thus resolving an open problem posed by Queyranne [135]. Our technique also shows that the number of optimal solutions is polynomial.

All of our results build on the random contraction approach of Karger [16]. Our techniques illustrate the versatility of the random contraction approach to address counting and algorithmic problems concerning multiobjective min-cuts and size-constrained k-cuts in hypergraphs. The results in this chapter are based on joint work with Chandrasekaran and Xu and appeared in Mathematical Programming in 2021 [83]. The research in this chapter was supported in part by NSF grants CCF-1814613 and CCF-1907937.

4.1 INTRODUCTION

Cuts and partitioning play a central role in combinatorial optimization and have numerous theoretical as well as practical applications. We consider multicriteria cut problems in hypergraphs. Let $G = (V, E)$ be an n-vertex hypergraph and $c_1, \ldots, c_t : E \to \mathbb{R}_{\geq 0}$ be t non-negative hyperedge-cost functions, where t is a fixed constant and the hypergraph induced by hyperedges of positive c_t cost is connected. The cost of a set of hyperedges $F \subseteq E$ under criterion $i \in [t]$ is $c_i(F) := \sum_{e\in F} c_i(e)$. For a partition $X = (X_1, \ldots, X_k)$ of V, we use $\delta(X)$ to denote the set of hyperedges that intersect at least two parts of X. For a subset U of vertices, we will use $\overline{U}$ to denote $V \setminus U$ and $\delta(U)$ to denote $\delta((U, \overline{U}))$. For a vertex v, we will use $\delta(v)$ to denote $\delta(\{v\})$. A subset F of hyperedges is a k-cut if there exists a partition $X = (X_1, \ldots, X_k)$ of V such that $F = \delta(X)$. We refer to a 2-cut simply as a cut. We recall that the rank of a hypergraph G is the size of the largest hyperedge in G (the rank of a graph is 2).

Since we have several criteria, there may not be a single cut that is best for all criteria. In multicriteria optimization, there are three important notions to measure the quality of a cut: (i) parametric min-cuts, (ii) pareto-optimal cuts, and (iii) multiobjective min-cuts. We define these notions now.

Definition 4.1. A cut F is a *parametric min-cut* if there exist multipliers $\mu_1, \ldots, \mu_t \in \mathbb{R}_+$ such that F is a min-cut in the hypergraph G with hyperedge costs given by $c_\mu(e) := \sum_{i=1}^{t} \mu_i c_i(e)$ for all $e \in E$.

Definition 4.2. A cut F *dominates* another cut F' if $c_i(F) \leq c_i(F')$ for every $i \in [t]$ and there exists $i \in [t]$ such that $c_i(F) < c_i(F')$. A cut F is *pareto-optimal* if it is not dominated by any other cut.

Definition 4.3. For a budget-vector $b \in \mathbb{R}^{t-1}_{\geq 0}$, a cut F is a *b-multiobjective min-cut* if $c_i(F) \leq b_i$ for every $i \in [t-1]$ and $c_t(F)$ is minimum subject to these constraints. A cut F is a *multiobjective min-cut* if there exists a non-negative budget-vector $b \in \mathbb{R}^{t-1}_{\geq 0}$ for which F is a b-multiobjective min-cut.

These three notions satisfy the following relationship with the containment being possibly strict (see Section 4.5 for a proof):

$$\text{Parametric min-cuts} \subseteq \text{Pareto-optimal cuts} \subseteq \text{Multiobjective min-cuts}. \tag{4.1}$$

There is also a natural notion of min-cuts under node-weighted budget constraints. Let $w_1, \ldots, w_t : V \to \mathbb{R}_{\geq 0}$ be vertex-weight functions and $c : E \to \mathbb{R}_+$ be a hyperedge-cost function. For a budget-vector $b \in \mathbb{R}^{t}_{\geq 0}$, a subset $F \subseteq E$ of hyperedges is a *node-budgeted b-multiobjective min-cut* if $F = \delta(U)$ for some subset $\emptyset \neq U \subsetneq V$ with $\sum_{u\in U} w_i(u) \leq b_i$ for all $i \in [t]$ and $c(F)$ is minimum among all such subsets of E. A cut F is a *node-budgeted multiobjective min-cut* if there exists a non-negative budget-vector b for which F is a node-budgeted b-multiobjective min-cut. In this chapter, we address the following natural questions concerning multiobjective min-cuts and min-k-cuts:

1. Multiobjective min-cuts: Is the number of multiobjective min-cuts at most strongly polynomial?

2. Node-budgeted multiobjective min-cuts: Is the number of node-budgeted multiobjective min-cuts at most strongly polynomial?

3. Size-constrained min-k-cut: For fixed positive integers k and $s_1, \ldots, s_k$ (all constants), a vertex-weight function $w : V \to \mathbb{Z}_+$, and a hyperedge-cost function $c : E \to \mathbb{R}_+$, can we compute a k-cut F with minimum $c(F)$ subject to the constraint that F is the set of hyperedges crossing some k-partition $(U_1, \ldots, U_k)$ of V where $\sum_{u \in U_i} w(u) \geq s_i$ for every $i \in [k]$ in polynomial-time? Is the number of optimal solutions strongly polynomial?

Previous work. For single criterion, a classic result of Dinitz, Karzanov, and Lomonosov [15] shows that the number of min-cuts in an n-vertex graph is $O(n^2)$ (also see Karger [16]). The same upper bound was shown to hold for constant-rank hypergraphs by Kogan and Krauthgamer [20] and for arbitrary-rank hypergraphs by Chekuri and Xu [38] and by Ghaffari, Karger, and Panigrahi [34] via completely different techniques. For $t = 2$ criteria in graphs, Mulmuley [136] showed an $O(n^{19})$ upper bound on the number of parametric min-cuts. For t criteria in constant-rank hypergraphs, Aissi, Mahjoub, McCormick, and Queyranne [81] showed that the number of parametric min-cuts is $\tilde{O}(m^t n^2)$, where m is the number of hyperedges, using the fact that the number of approximate min-cuts in constant-rank hypergraphs is polynomial. Karger [82] improved this bound to $O(n^{t+1})$ by a clever and subtle argument based on his random contraction algorithm; we will describe his argument later. Karger also constructed a graph that exhibited $\Omega(n^{t/2})$ parametric min-cuts.

Armon and Zwick [78] showed that all pareto-optimal cuts in graphs can be enumerated in pseudo-polynomial time. For $t = 2$ criteria in constant-rank hypergraphs, Aissi, Mahjoub, McCormick, and Queyranne [81] showed an upper bound of $\tilde{O}(n^5)$ on the number of pareto-optimal cuts—this was the first result showing a strongly polynomial upper bound. These authors raised the question of whether the number of pareto-optimal cuts is strongly polynomial for a constant number t of criteria in constant-rank hypergraphs (or even in graphs). Note that, by containment relationship (4.1), answering our first question affirmatively would also answer their open question. On a related note, Aissi, Mahjoub, and Ravi [79] designed a random contraction based algorithm to solve the b-multiobjective min-cut problem in graphs. The correctness analysis of their algorithm also implies that the number of b-multiobjective min-cuts in graphs for a fixed budget-vector $b \in \mathbb{R}_+^{t-1}$ is $O(n^{2t})$.

We emphasize the subtle, but important, distinction between the number of b-multiobjective min-cuts for a fixed budget-vector b and the number of multiobjective min-cuts.

Node-budgeted multiobjective min-cut has a rich literature extending nicely to submodular functions. For graphs, Armon and Zwick [78] gave a polynomial-time algorithm to find a minimum valued cut with at most b vertices in the smaller side. Goemans and Soto [137] addressed the more general problem of minimizing a symmetric submodular function $f : 2^V \to \mathbb{R}$ over a downward-closed family $\mathcal{I}$. Recall that the hypergraph cut function is symmetric submodular and the family of vertex subsets satisfying node-weighted budget constraints is in fact downward-closed. Goemans and Soto extended Queyranne's submodular minimization algorithm to enumerate all the $O(n)$ minimal minimizers in $\mathcal{I}$ using $O(n^3)$ oracle calls to the function f and the family $\mathcal{I}$. Their result implies that the number of minimal minimizers is $O(n)$, but it is straightforward to see that the total number of minimizers could be exponential. For the special case of node-budgeted multiobjective min-cuts in graphs, Aissi, Mahjoub, and Ravi [79] gave a faster algorithm than that of Goemans and Soto—their algorithm is based on random contraction, runs in $\tilde{O}(n^2)$-time, and shows that the number node-budgeted b-multiobjective min-cuts in graphs for a fixed budget-vector $b \in \mathbb{R}^t_{\geq 0}$ is $O(n^2)$.

For size-constrained min-k-cut, if we allow arbitrary sizes (i.e., arbitrary lower bounds), then the problem becomes NP-hard even for $k = 2$ as it captures the well-studied min-bisection problem in graphs. If we consider constant sizes but arbitrary k, then the problem is again NP-hard in graphs [72]. So, our focus is on constant k and constant sizes. Guinez and Queyranne [135] raised size-constrained min-k-cut with unit vertex-weights as a subproblem towards resolving the complexity of the submodular k-partitioning problem. In submodular k-partitioning, we are given a submodular function $f : 2^V \to \mathbb{R}$ (by value oracle) and a fixed constant integer k (e.g., $k = 2, 3, 4, 5, \ldots$) and the goal is to find a k-partition $(U_1, \ldots, U_k)$ of the ground set V so as to minimize $\sum_{i=1}^{k} f(U_i)$. The complexity of even special cases of this problem are open: e.g., if the submodular function f is the cut function of a given hypergraph, then its complexity is unknown.[6] Guinez and Queyranne showed surprisingly strong non-crossing properties between optimum solutions to size-constrained $(k-1)$-partitioning (constant size lower bounds on the parts) and optimum solutions to k-partitioning. This motivated them to study the size-constrained min-k-cut problem in hypergraphs for unit vertex-weights as a special case. They showed that size-constrained

[6]We note that if the submodular function f is the cut function of a given hypergraph, then the submodular k-partition problem is not identical to hypergraph k-cut as the two objectives are different. However, if the submodular function is the cut function of a given graph, then the submodular k-partition problem coincides with the graph k-cut problem which is solvable in polynomial-time.

min-k-cut for unit vertex-weights is solvable in polynomial-time in constant-rank hypergraphs (with exponential run-time dependence on the rank) and mention the open problem of designing an algorithm for it in arbitrary-rank hypergraphs. The size-constrained min-k-cut problem for unit sizes (i.e., all size lower-bounds $s_1, \ldots, s_k$ are equal to one) is known as the hypergraph k-cut problem. The hypergraph k-cut problem was shown to admit a polynomial-time algorithm only recently [138] via a non-uniform random contraction algorithm.

4.1.1 Our Contributions

Our high-level contribution is in showing the versatility of the random contraction technique to address algorithmic and counting problems concerning multiobjective min-cuts and size-constrained min-k-cuts in hypergraphs. All of our results build on the random contraction technique with additional insights.

Our first result is a strongly polynomial upper bound on the number of multiobjective min-cuts in constant-rank hypergraphs.

Theorem 4.1. The number of multiobjective min-cuts in an r-rank, n-vertex hypergraph with t hyperedge-cost functions is $O(r2^{rt}n^{3t-1})$.

We emphasize that our upper bound is over all possible non-negative budget-vectors (in contrast to the number of b-multiobjective min-cuts for a fixed budget-vector b). Theorem 4.1 and containment relationship (4.1) imply that the number of pareto-optimal cuts in constant-rank hypergraphs is $O(n^{3t-1})$ and hence, is strongly polynomial for constant number of criteria. This answers the main open question posed by Aissi, Mahjoub, McCormick, and Queyranne [81]. We also design randomized polynomial time algorithms to enumerate all multiobjective min-cuts and all pareto-optimal cuts in constant-rank hypergraphs (see Section 4.2.3). Independent of our work, Rico Zenklusen has also shown Theorem 4.1 using a different approach. We learned after the conference submission of this work that his approach also leads to *deterministic* polynomial time algorithms to enumerate all multiobjective min-cuts and all pareto-optimal cuts in constant-rank hypergraphs. We will outline his proof approach in the technical section.

Given the upper bound in Theorem 4.1, a discussion on the lower bound is in order. We recall that Karger [82] constructed a graph with t edge-cost functions that exhibited $\Omega(n^{t/2})$ parametric min-cuts. This is also a lower bound on the number of pareto-optimal cuts and multiobjective min-cuts by (4.1). We improve this lower bound for pareto-optimal cuts by constructing a graph with t edge-cost functions that exhibits $\Omega(n^t)$ pareto-optimal cuts (see Section 4.2.4). Our instance also exhibits the same lower bound on the number of b-multiobjective min-cuts for a fixed budget-vector b.

Our next result is an upper bound on the number of node-budgeted multiobjective min-cuts and node-budgeted b-multiobjective min-cuts.

Theorem 4.2. Let G be an r-rank, n-vertex hypergraph with t vertex-weight functions. Then, the following two hold:

1. The number of node-budgeted multiobjective min-cuts in G is $O(r2^r n^{t+2})$.

2. For a fixed budget-vector $b \in \mathbb{R}^t_{\geq 0}$, the number of node-budgeted b-multiobjective min-cuts in G is $O(n^2)$.

We draw the reader's attention to the distinction between the two parts in Theorem 4.2. The first part implies that the number of node-budgeted multiobjective min-cuts is strongly polynomial in constant-rank hypergraphs for constant number of vertex-weight functions. The second part implies that the number of node-budgeted b-multiobjective min-cuts for any fixed budget-vector $b \in \mathbb{R}^t_{\geq 0}$ is strongly polynomial in arbitrary-rank hypergraphs for any number t of vertex-weight functions.

Our final result shows that the size-constrained min-k-cut problem can be solved in polynomial time for constant k and constant sizes (in arbitrary-rank hypergraphs).

Theorem 4.3. Let $k \geq 2$ be a fixed positive integer and let $1 \leq s_1 \leq s_2 \leq \ldots \leq s_k$ be fixed positive integers. Let $G = (V, E)$ be an n-vertex hypergraph with hyperedge-cost function $c : E \to \mathbb{R}_+$. Then, there exists a polynomial-time algorithm that takes (G, c) as input and returns a fixed w-weighted s-size-constrained min-k-cut for any choice of vertex-weight function $w : V \to \mathbb{Z}_+$ with probability

$$\Omega\left(\frac{1}{n^{2\sigma_{k-1}+1}}\right),$$

where $\sigma_{k-1} := \sum_{i=1}^{k-1} s_i$.

Theorem 4.3 resolves an open problem posed by Guinez and Queyranne [135]. A structural consequence of Theorem 4.3 is that the number of size-constrained min-k-cuts (over all possible node-weight functions $w : V \to \mathbb{Z}_+$) in a given hypergraph is polynomial for constant sizes and constant k.

We refer the reader to Table 4.1 for a comparison of known results and our contributions.

Problem	**Graphs**	**r-rank Hypergraphs**	**Hypergraphs**
# of Parametric Min-Cuts	$O(n^{t+1})$ [82] $\Omega(n^{t/2})$ [82]	$O(2^r n^{t+1})$ [82]	OPEN
# of Pareto-Optimal Cuts	$\tilde{O}(n^5)$ for $t = 2$ [81] $O(n^{3t-1})$ [Thm 4.1] $\Omega(n^t)$ [Thm 4.7]	$O(r2^{rt}n^{3t-1})$ [Thm 4.1]	OPEN
# of b-Multiobjective Min-Cuts	$O(n^{2t})$ [79] $\Omega(n^t)$ [Thm 4.7]	$O(r2^{rt}n^{2t})$ [Thm 4.4]	OPEN
# of Multiobjective Min-Cuts	$O(n^{3t-1})$ [Thm 4.1] $\Omega(n^t)$ [Thm 4.7]	$O(r2^{rt}n^{3t-1})$ [Thm 4.1]	OPEN
# of Node-Budgeted b-Multiobjective Min-Cuts	$O(n^2)$ [79]	$O(n^2)$ [Thm 4.2]	$O(n^2)$ [Thm 4.2]
# of Node-Budgeted Multiobjective Min-Cuts	$O(n^{t+2})$ [Thm 4.2]	$O(r2^r n^{t+2})$ [Thm 4.2]	OPEN
Node-Weighted s-Size-Constrained k-cut (const. k and const. $s \in \mathbb{Z}^k$)	Poly-time [135]	Poly-time [135]	Poly-time [Thm 4.3]

Table 4.1: Here, t denotes the number of criteria (i.e., the number of hyperedge-cost/vertex-weight functions), r denotes the rank of the hypergraph, and n denotes the number of vertices.

4.1.2 Technical Overview

As mentioned earlier, all our results build on the random contraction technique introduced by Karger [16] to solve the global min-cut problem in graphs. Here, a *uniform* random edge of the graph is contracted in each step until the graph has only two nodes; the set of edges between the two nodes is returned as the cut. Karger showed that this algorithm returns a fixed global min-cut with probability $\Omega(n^{-2})$. As a consequence, the number of min-cuts in an n-vertex graph is $O(n^2)$. The algorithm extends naturally to r-rank hypergraphs, however the naive analysis will only show that the algorithm returns a fixed global min-cut with probability $\Omega(n^{-r})$. Kogan and Krauthgamer [20] introduced an LP-based analysis thereby showing that the algorithm indeed succeeds with probability $\Omega(2^{-r}n^{-2})$. As a consequence, the number of global min-cuts in constant-rank hypergraphs is also $O(n^2)$.

In a recent work, Karger observed that uniform random contraction can also be used to bound the number of parametric min-cuts in constant-rank hypergraphs. We describe his argument for graphs since two of our theorems build on it. Suppose we fix the multipliers $\mu_1, \ldots, \mu_t$ in the parametric min-cut problem, then a fixed min c_μ-cost cut can be obtained with probability $\Omega(n^{-2})$ by running the random contraction algorithm with respect to the edge-cost function c_μ. Karger suggested an alternative viewpoint of the execution of the algorithm for the edge-cost function c_μ. For simplicity, we assume parallel edges instead of costs, i.e., $c_i(e) \in \{0, 1\}$ for every edge e and every criterion $i \in [t]$. Let E_i be the set of edges with non-zero weight in the i'th criterion. The execution of the random contraction algorithm wrt c_μ can alternatively be specified as follows: a permutation π_i of the edges in E_i for each $i \in [t]$ and an *interleaving* indicating at each step whether the algorithm contracts the next edge from π_1 or π_2 or $\ldots$ or π_t. Critically, the sequences π_i for every $i \in [t]$ can be assumed to be uniformly random. Thus, we can move all randomness upfront, namely pick a uniform random permutation π_i for each criterion $i \in [t]$. Now, instead of returning one cut, we return the collection of cuts produced by contracting along all possible interleavings. This modified algorithm no longer depends on the specific multipliers $\mu_1, \ldots, \mu_t$ and hence, a parametric min-cut for any fixed choice of multipliers $\mu_1, \ldots, \mu_t$ will be in the output collection with probability at least $\Omega(n^{-2})$. It remains to bound the number of interleavings since that determines the number of cuts in the returned collection: the crucial observation here is that the number of interesting interleavings is only n^{t-1}. This is because interleaved contractions produce the same final graph as performing a certain number of contractions according to π_1 (until the number of vertices is, say, n_1), then a certain number of contractions based on π_2 (until the number of vertices is, say, n_2), and so on. So, the order of contractions becomes irrelevant and only the numbers of vertices $n_1, \ldots, n_t$ are relevant. Overall, this implies that the number of parametric min-cuts is $O(n^{t+1})$. We emphasize that this *interleaving argument* relies crucially on the basic random contraction algorithm picking edges to contract according to a *uniform distribution* (allowing the permutations $\pi_1, \ldots, \pi_t$ to be uniform random permutations).

Next, we describe our approach underlying the proof of Theorem 4.1, but for graphs. In order to bound the number of multiobjective min-cuts through the interleaving argument, we first need a random contraction based algorithm to solve the b-multiobjective min-cut problem. Indeed, Aissi, Mahjoub, and Ravi [79] designed a random contraction based algorithm to solve the b-multiobjective min-cut problem in graphs. Their algorithm proceeds as follows: For each $i \in [t-1]$, let U_i be the set of vertices $u \in V - \cup_{j=1}^{i-1} U_j$ for which $c_i(\delta(u)) > b_i$ (known as the set of i-infeasible vertices), and let $U_t := V - \cup_{j=1}^{t-1} U_j$. In each step, they pick $i \in [t]$ with probability proportional to the number of i-infeasible vertices (i.e., $|U_i|$) and pick a random edge e among the ones incident to U_i with probability proportional to $c_i(e)$, contract e, and repeat. Unfortunately, this algorithm does not have the *uniform distribution* that is crucially necessary to apply Karger's interleaving argument. To introduce uniformity to the distribution, we modify this algorithm in two ways:

1. At each step we deterministically choose the criterion i corresponding to the largest U_i (as opposed to picking i randomly with probability proportional to $|U_i|$).

2. Next, we choose a uniform random edge e from among all edges in the graph with probability proportional to $c_i(e)$ (as opposed to picking an edge only from among the edges incident to U_i). We contract this chosen edge e.

These two features bring a *uniform distribution* property to the algorithm, which in turn, allows us to apply the interleaving argument. With these two features, we show that the algorithm returns a fixed b-multiobjective min-cut for a fixed budget-vector b with probability $\Omega(n^{-2t})$. Armed with the two features, we move all randomness upfront using the interleaving argument. As a consequence, we obtain that the total number of multiobjective min-cuts (irrespective of the choice of budget-vector b) is $O(n^{3t-1})$. For constant-rank hypergraphs, we perform an LP-based analysis of our algorithm for b-multiobjective min-cut (thus, extending Kogan and Krauthgamer's analysis) to arrive at the same success probability. The interleaving argument for constant-rank hypergraphs proceeds similarly.

We emphasize that the interleaving argument does not extend to arbitrary-rank hypergraphs. This is because the currently known random contraction based algorithms for min-cut in arbitrary-rank hypergraphs crucially require non-uniform contractions (the next hyperedge to contract is chosen from a distribution that depends on the current sizes of all hyperedges), so we cannot assume that the permutations $\pi_1, \pi_2, \ldots, \pi_t$ are uniformly random. Consequently, we do not even know if the number of parametric min-cuts in a hypergraph is at most strongly polynomial. Another interesting open question here is whether the b-multiobjective min-cut problem in hypergraphs is solvable in polynomial-time even for $t = 2$ criteria. We have arrived at hypergraph instances (with large rank) for which Aissi, Mahjoub, and Ravi's approach (as well as our modified approach) will never succeed, even with non-uniform random contractions.

Next, we outline the proof of Theorem 4.2. The approach is to again design a random contraction algorithm that returns a fixed node-budgeted b-multiobjective min-cut with probability $\Omega(n^{-2})$ (in both constant-rank and arbitrary-rank hypergraphs). Such an algorithm would imply the second part of the theorem immediately while the first part would follow if we can apply an interleaving-like argument (i.e., the designed algorithm performs uniform random contractions). Our approach is essentially an extension of the approach by Goemans and Soto who suggested contracting the infeasible vertices together (a vertex u is infeasible if $w_i(u) > b_i$ for some $i \in [t]$). Aissi, Mahjoub, and Ravi show that doing this additional step after each random contraction step returns a fixed node-budgeted b-multibjective min-cut with probability $\Omega(n^{-2})$ in graphs. Our main contribution is showing that this additional "contracting infeasible vertices together" step in conjunction with (1) uniform random contractions for constant-rank hypergraphs and (2) non-uniform random contractions for arbitrary hypergraphs succeeds with the required probability. Next, a naive interleaving-like argument can be applied for constant-rank hypergraphs to conclude that the number of node-budgeted multiobjective min-cuts is $O(n^{t+3})$. We improve this to $O(n^{t+2})$ with a more careful argument.

Finally, we outline our approach for Theorem 4.3. Guinez and Queyranne address size-constrained k-cut in constant-rank hypergraphs for unit vertex-weights by performing uniform random contractions until the number of nodes in the hypergraph is close to $\sum_{i=1}^{k} s_i$ at which point they return a uniform random cut. Their success probability has exponential dependence on the rank. The key technical ingredient to bring down the exponential dependence on rank is the use of non-uniform contractions. For the special case of unit sizes and unit vertex-weights (i.e., the hypergraph k-cut problem), Chandrasekaran, Xu, and Yu [138] introduced an explicit non-uniform distribution that leads to a success probability of $\Omega(n^{-2(k-1)})$. Our algorithm extends the non-uniform distribution to arbitrary but constant sizes (as opposed to just unit sizes), yet without depending on vertex-weights. Our analysis takes care of the vertex-weight function through weight tracking, i.e., by declaring the weight of a contracted node to be the sum of the weight of the vertices in the hyperedge being contracted. We note that our algorithm's success probability when specialized to the case of unit vertex-weights and unit sizes is weaker than the success probability of the algorithm by Chandrasekaran, Xu, and Yu (by a factor of n). We leave it as an open question to improve this. On the other hand, our algorithm has the added advantage that it does not even take the vertex-weight function w as input and yet succeeds in returning a w-vertex-weighted s-size-constrained k-cut for any choice of w with inverse polynomial probability.

Organization. In Section 4.2, we bound the number of multiobjective min-cuts (and prove Theorem 4.1), give efficient algorithms to enumerate all multiobjective min-cuts and all pareto-optimal cuts, and present lower bounds on the number of pareto-optimal cuts and the number of b-multiobjective min-cuts. In Section 4.3, we address node-budgeted multiobjective min-cuts and prove Theorem 4.2. In Section 4.4, we give an algorithm to solve size-constrained min-k-cut, thereby proving Theorem 4.3. In Section 4.5 we show some relationships between the different kinds of multicriteria cuts that we consider. In Section 4.6 we conclude by discussing some open problems.

4.1.3 Preliminaries

We define the random contraction procedure that is central to all of our algorithms. Let $G = (V, E)$ be a hypergraph, and let $U \subseteq V$ be a set of vertices in G. We define G *contract* U, denoted G/U, to be a hypergraph on the vertex set $(V \setminus U) \cup \{u\}$, where u is a newly introduced vertex. The hyperedges of G/U are defined as follows. For each hyperedge $e \in E$ of G, such that $e \not\subseteq U$, G/U has a corresponding hyperedge e', where $e' := e$ if $e \subseteq V \setminus U$ and $e' := (e \setminus U) \cup \{u\}$ otherwise, and $c_i(e') = c_i(e)$ for every $i \in [t]$. If w is a vertex-weight function for G, then we will also use w as a vertex-weight function for G/U. We define the weight of the newly introduced vertex u as $w(u) := \sum_{v \in U} w(v)$.

We will need the following lemma that will be used in the analysis of two of our algorithms.

Lemma 4.1. Let r, γ, n be positive integers with $n \geq \gamma \geq r + 1 > 2$. Let $f : \mathbb{N} \to \mathbb{R}_+$ be a positive-valued function defined over the natural numbers. Then, the optimum value of the linear program defined below is $\min_{2 \leq j \leq r}(1 - \frac{j}{\gamma - r + j})f(n - j + 1)$.

$$\begin{aligned}
\underset{x_2,\ldots,x_r,y_2,\ldots,y_r}{\text{minimize}} \quad & \sum_{j=2}^{r}(x_j - y_j)f(n-j+1) & \\
\text{subject to} \quad & 0 \leq y_j \leq x_j & \forall j \in \{2, \ldots, r\} \\
& \sum_{j=2}^{r} x_j = 1 & \\
& \gamma \sum_{j=2}^{r} y_j \leq \sum_{j=2}^{r} j \cdot x_j &
\end{aligned}$$

Proof. Setting $x_2 = 1$ and the rest of the variables to zero gives a feasible solution to the linear program (LP). Thus, the LP is feasible. Let $j \in \{2, \ldots, r\}$. Since $y_j \geq 0$ and $x_j \leq 1$, we have that $x_j - y_j \leq 1$. Since f is positive valued, we have that $f(n - j + 1) \geq 0$ for every $j \in [2, r]$, so it follows that $(x_j - y_j)f(n - j + 1) \leq f(n - j + 1)$. Therefore, we have $\sum_{j=2}^{r}(x_j - y_j)f(n - j + 1) \leq \sum_{j=2}^{r} f(n - j + 1)$. Thus, the objective value of this LP is bounded. Since the LP is feasible and bounded, there exists an extreme point optimum solution to this LP. The LP has $2r - 2$ variables and $2r$ equations, so every extreme point optimum will have at least $2r - 2$ tight constraints and at most 2 non-tight constraints.

We now show that the final constraint of the LP is tight for every optimal solution (x, y). Let (x, y) be an optimal solution. We first note that $x_j > 0$ for some $j \in \{2, \ldots, r\}$, by the third constraint. Therefore it is impossible that $y_j = 0$ for every $j \in \{2, \ldots, r\}$, since, if this were the case, we could choose some $j \in \{2, \ldots, r\}$ such that $x_j > 0$ and then increase

y_j by a small amount to improve the value of the objective function without violating any constraints, contradicting optimality. Now, since $\gamma \geq r+1$, we have

$$\gamma \sum_{j=2}^{r} y_j \geq \sum_{j=2}^{r} (r+1) y_j > \sum_{j=2}^{r} j \cdot y_j. \tag{4.2}$$

This implies that we cannot have $y_j = x_j$ for all j, otherwise (x, y) would violate the final constraint of the LP. Hence, at least one of the $y_j \leq x_j$ constraints must be slack. Let $j \in \{2, \ldots, r\}$ be such that $y_j < x_j$. If the final constraint was slack, increasing the value of y_j by a very small amount would improve the objective value of (x, y) without violating any constraints. Therefore, since (x, y) is optimal, the final constraint must be tight.

Let (x, y) be an extreme point optimal solution. Since we know that $\sum_{j=2}^{r} x_j = 1$ and $x_j \geq 0$ for every $j \in \{2, \ldots, r\}$, we have $\sum_{j=2}^{r} j \cdot x_j > 0$. Since the final constraint is tight for (x, y), we must have $\gamma \sum_{j=2}^{r} y_j > 0$. This implies that there exists $j \in [2, r]$ such that $y_j > 0$. Thus, we conclude that the two slack constraints must be $0 \leq y_{j_1}$ and $y_{j_2} \leq x_{j_2}$ for some $j_1, j_2 \in \{2, \ldots, r\}$. We consider two cases.

Case 1: Suppose $j_1 = j_2$. Then we have that $0 < y_j < x_j = 1$ for some $j \in \{2, \ldots, r\}$, and $x_{j'}, y_{j'} = 0$ for every $j' \in \{2, \ldots, r\} \setminus \{j\}$. Therefore, we can simplify our LP to

$$\begin{aligned} \underset{y_j}{\text{minimize}} \quad & (1 - y_j) f(n - j + 1) && (4.3) \\ \text{subject to} \quad & 0 \leq y_j \leq 1 && (4.4) \\ & \gamma y_j = j. && (4.5) \end{aligned}$$

The only (and therefore optimal) solution to this LP is $y_j = \frac{j}{\gamma}$, which achieves an objective value of

$$\left(1 - \frac{j}{\gamma}\right) f(n - j + 1). \tag{4.6}$$

Case 2: Suppose $j_1 \neq j_2$. Then we have that $0 < y_{j_1} = x_{j_1}$, and $0 = y_{j_2} < x_{j_2}$. We note that $x_{j_2} = 1 - x_{j_1}$, and therefore we can simplify the LP to

$$\begin{aligned} \underset{x_{j_1}}{\text{minimize}} \quad & (1 - x_{j_1}) f(n - j_2 + 1) && (4.7) \\ \text{subject to} \quad & 0 \leq x_{j_1} \leq 1 && (4.8) \\ & \gamma x_{j_1} = j_1 \cdot x_{j_1} + j_2 \cdot (1 - x_{j_1}). && (4.9) \end{aligned}$$

Solving the second constraint for x_{j_1} yields $x_{j_1} = \frac{j_2}{\gamma - j_1 + j_2}$, and therefore our optimum value is

$$\left(1-\frac{j_2}{\gamma-j_1+j_2}\right)f(n-j_2+1). \tag{4.10}$$

We conclude that the optimum value of the LP is equal to the minimum of the values from these two cases, that is,

$$\min\left\{\min_{j\in\{2,\ldots,r\}}\left\{\left(1-\frac{j}{\gamma}\right)f(n-j+1)\right\},\ \min_{j_1,j_2\in\{2,\ldots,r\}}\left\{\left(1-\frac{j_2}{\gamma-j_1+j_2}\right)f(n-j_2+1)\right)\right\}\right\}. \tag{4.11}$$

Since $(1-\frac{j_2}{\gamma-j_1+j_2})$ is decreasing in j_1 and $f(n-j_2+1)$ is always positive, we have

$$\min_{j_1,j_2\in\{2,\ldots,r\}}\left\{\left(1-\frac{j_2}{\gamma-j_1+j_2}\right)f(n-j_2+1)\right\}=\min_{j\in\{2,\ldots,r\}}\left\{\left(1-\frac{j}{\gamma-r+j}\right)f(n-j+1)\right\}. \tag{4.12}$$

Thus, since $j\le r$, the optimum value of the LP is equal to

$$\min_{j\in\{2,\ldots,r\}}\left\{\min\left\{1-\frac{j}{\gamma},1-\frac{j}{\gamma-r+j}\right\}f(n-j+1)\right\} \tag{4.13}$$

$$=\min_{j\in\{2,\ldots,r\}}\left\{\left(1-\frac{j}{\gamma-r+j}\right)f(n-j+1)\right\}. \tag{4.14}$$

QED.

4.2 MULTIOBJECTIVE MIN-CUTS AND PARETO-OPTIMAL CUTS

In this section, we give upper and lower bounds on the number of multiobjective min-cuts and pareto-optimal cuts and prove Theorem 4.1 (see Definitions 4.3 and 4.2).

Let $G=(V,E)$ be an r-rank hypergraph and let $c_1,\ldots,c_t:E\to\mathbb{R}_{\ge 0}$ be cost functions on the hyperedges of G such that the hypergraph induced by hyperedges of positive c_t cost is connected. We note that the assumption that the hypergraph induced by hyperedges of positive c_t cost is connected is required to say that the number of b-multiobjective min-cuts in G is polynomial, since a complete graph where every edge has zero c_t cost will have exponentially many b-multiobjective min-cuts for sufficiently large b.

We begin with a randomized algorithm for the b-multiobjective min-cut problem in Section 4.2.1. We take an alternative viewpoint of this randomized algorithm in Section 4.2.2 to prove Theorem 4.1. Since all pareto-optimal cuts are multiobjective min-cuts, Theorem 4.1 also implies that the number of pareto-optimal cuts in an r-rank n-vertex hypergraph G with t hyperedge-cost functions is $O(r2^{rt}n^{3t-1})$. In Section 4.2.3, we give randomized polynomial-time algorithms to enumerate all pareto-optimal cuts and all multiobjective min-cuts. In

Section 4.2.4, we show a lower bound of $\Omega(n^t)$ on the number of pareto-optimal cuts and on the number of b-multiobjective min-cuts.

4.2.1 Finding b-Multiobjective Min-Cuts

In this section, we design a randomized algorithm for the b-multiobjective min-cut problem, which is formally defined below.

b-MULTIOBJECTIVE MIN-CUT

Given: A hypergraph $G = (V, E)$ with hyperedge-cost functions $c_1, \ldots, c_t : E \to \mathbb{R}_{\geq 0}$, such that the hypergraph induced by hyperedges of positive c_t cost is connected, and a budget-vector $b \in \mathbb{R}_{\geq 0}^{t-1}$.

Goal: A b-multiobjective min-cut.

Problem 4.1: b-Multiobjective Min-Cut

We use Algorithm 4.1. We summarize its correctness and run-time guarantees in Theorem 4.4.

```
b-MULTIOBJECTIVE-MIN-CUT($G, r, t, c, b$):
    Input:  An $r$-rank hypergraph $G = (V, E)$, hyperedge-cost functions
            $c_1, \ldots, c_t : E \to \mathbb{R}_{\geq 0}$ and a budget-vector $b \in \mathbb{R}_{\geq 0}^{t-1}$.
    If $|V| \leq rt$:
        $X \leftarrow$ a random subset of $V$
        return $\delta(X)$
    For $i = 1, \ldots, t-1$:
        $U_i \leftarrow \{v \in V \setminus (\bigcup_{j=1}^{i-1} U_j) : c_i(\delta(v)) > b_i\}$
    $U_t \leftarrow V \setminus \bigcup_{j=1}^{t-1} U_j$
    $i \leftarrow \arg\max_{j \in [t]} |U_j|$
    $e \leftarrow$ a random hyperedge chosen according to $\Pr[e = e'] = \frac{c_i(e')}{c_i(E)}$
    $G' \leftarrow G/e$
    Return b-MULTIOBJECTIVE-MIN-CUT($G', r, t, c, b$)
```

Algorithm 4.1: b-MULTIOBJECTIVE MIN-CUT

Theorem 4.4. Let $G = (V, E)$ be an r-rank n-vertex hypergraph with hyperedge-cost functions $c_1, \ldots, c_t : E \to \mathbb{R}_{\geq 0}$, such that the hypergraph induced by hyperedges of positive c_t cost is connected, and let $b \in \mathbb{R}_{\geq 0}^{t-1}$ be a budget-vector. Let F be an arbitrary b-multiobjective min-cut. Then, Algorithm 4.1 outputs F with probability at least Q_n, where

$$Q_n := \begin{cases} \frac{1}{2^{rt}} & \text{if } n \leq rt, \textit{and} \\ \frac{2t+1}{2^{rt}(rt+1)}\binom{n-t(r-2)}{2t}^{-1} & \text{if } n > rt. \end{cases}$$

Moreover, the algorithm can be implemented to run in polynomial time.

Proof. We note that the sets U_i can be computed in polynomial time, and the algorithm recomputes them at most n times. Random contraction can also be implemented in polynomial time, and therefore the overall run-time of the algorithm is polynomial. We also note that whenever we contract a hyperedge, $c_i(E) > 0$. This is because, if $i < t$, then in order for U_i to be the largest of $U_1, \ldots, U_t$, some vertex v must have $c_i(\delta(v)) > b_i \geq 0$, and if $i = t$, then $c_t(E) > 0$ since the hypergraph induced by hyperedges of positive c_t cost is connected.

We now bound the correctness probability by induction on n. For the base case, we consider $n \leq rt$. In this case, the algorithm returns $\delta(X)$ for a random $X \subseteq V$. There are 2^n possible values for X, and $F = \delta(X)$ for at least one of them. Thus, the probability that the algorithm returns F is at least $\frac{1}{2^n} \geq \frac{1}{2^{rt}} = Q_n$.

Next, we prove the induction step. Let $n > rt$ and assume that the theorem holds for all hypergraphs with at most $n-1$ vertices and rank at most r. We will need the following claim.

Claim 4.1. The algorithm outputs F with probability at least the optimum value of the following linear program.

$$\begin{aligned} \underset{x_2,\ldots,x_r,y_2,\ldots,y_r}{\text{minimize}} \quad & \sum_{j=2}^{r}(x_j - y_j)Q_{n-j+1} \\ \text{subject to} \quad & 0 \leq y_j \leq x_j \ \forall j \in \{2,\ldots,r\} \\ & \sum_{j=2}^{r} x_j = 1 \\ & |U_i|\sum_{j=2}^{r} y_j \leq \sum_{j=2}^{r} j \cdot x_j \end{aligned}$$

Proof. Since $n > rt$, when the algorithm is executed on G it will contract a randomly chosen hyperedge and recurse. Let e' be the random hyperedge chosen by the algorithm. If $e' \notin F$, then F will still be a b-multiobjective min-cut in G/e'. We observe that G/e' is a hypergraph with $n - |e'| + 1$ vertices and the rank of G/e' is at most the rank of G. Therefore, if $e' \notin F$, then the algorithm will output F with probability at least $Q_{n-|e'|+1}$.

Let $i \in [t]$ be the index of the cost function chosen by the algorithm. Let

$$E_j := \{e \in E \colon |e| = j\}, \tag{4.15}$$

$$x_j := \Pr[e' \in E_j] = \frac{c_i(E_j)}{c_i(E)}, \text{ and} \tag{4.16}$$

$$y_j := \Pr[e' \in E_j \cap F] = \frac{c_i(E_j \cap F)}{c_i(E)}. \tag{4.17}$$

As we noted above, $c_i(E) > 0$ always, so these values are well-defined. We note that E_j is the set of hyperedges of size j, x_j is the probability of picking a hyperedge of size j to contract, and y_j is the probability of picking a hyperedge of size j from F to contract. We know that

$$\Pr[\text{Algorithm returns the cut } F] \geq \sum_{j=2}^{r} (x_j - y_j) Q_{n-j+1}. \tag{4.18}$$

The values of x_j and y_j will depend on the structure of G. However we can deduce some relationships between them. Since $0 \leq c_i(E_j \cap F) \leq c_i(E_j)$ for every $j \in \{2, \ldots, r\}$, we know that

$$0 \leq y_j \leq x_j \text{ for every } j \in \{2, \ldots, r\}. \tag{4.19}$$

Moreover, x_j is the probability of picking a hyperedge of size j. Hence,

$$\sum_{j=2}^{r} x_j = 1. \tag{4.20}$$

Next, we show that for every $i \in [t]$ and every $v \in U_i$, we have

$$c_i(F) \leq c_i(\delta(v)). \tag{4.21}$$

If $i < t$, then $c_i(F) \leq b_i < c_i(\delta(v))$ for every $v \in U_i$. Let $i = t$. We recall that F is a b-multiobjective min-cut. Since every cut induced by a single vertex in U_t satisfies all of the budgets, no such cut can have a better c_t-cost than F, so $c_t(F) \leq c_t(\delta(v))$ for every $v \in U_t$.

From inequality (4.21), we conclude that

$$c_i(F) \leq \frac{\sum_{v \in U_i} c_i(\delta(v))}{|U_i|} \leq \frac{\sum_{v \in V} c_i(\delta(v))}{|U_i|} = \frac{\sum_{e \in E} |e| c_i(e)}{|U_i|} = \frac{\sum_{j=2}^{r} j \cdot c_i(E_j)}{|U_i|}. \tag{4.22}$$

Therefore

$$\sum_{j=2}^{r} y_j = \Pr[e' \in F] = \frac{c_i(F)}{c_i(E)} \leq \frac{1}{|U_i|} \sum_{j=2}^{r} j \cdot x_j. \tag{4.23}$$

Thus, we have that

$$|U_i| \sum_{j=2}^{r} y_j \leq \sum_{j=2}^{r} j \cdot x_j. \tag{4.24}$$

The minimum value of our lower bound in equation (4.18) over all choices of x_j and y_j that satisfy inequalities (4.19), (4.20), and (4.24) is a lower bound on the probability that the algorithm outputs F. QED.

Let U_i be the largest among the sets $U_1, \ldots, U_t$ that the algorithm generates when executed on input (G, r, t, c, b). Claim 4.1 tells us that the algorithm outputs F with probability at least the optimum value of the linear program from the claim.

The linear program in Claim 4.1 is exactly the linear program from Lemma 4.1 with $\gamma = |U_i|$ and $f(n) := Q_n$. To apply Lemma 4.1, we just need to show that $n \geq |U_i| \geq r + 1$. We recall that U_i is the largest of the t sets that the algorithm constructs. Each of these sets is a subset of V, so we can conclude that $|U_i| \leq |V| = n$. We also know from the construction of the sets $U_1, \ldots, U_t$ that they partition V. This means that $\sum_{j=1}^{t} |U_j| = n$. Since U_i is the largest of the sets, we must have $|U_i| \geq \frac{n}{t}$. Since $n > rt$, this means $|U_i| \geq \frac{rt+1}{t} > r$. Thus $|U_i| > r$, and since r and $|U_i|$ are integers, we conclude that $|U_i| \geq r + 1$. Therefore, we can apply Lemma 4.1 with $\gamma = |U_i|$ to conclude that

$$\Pr[\text{Algorithm returns the cut } F] \geq \min_{2 \leq j \leq r} \left(1 - \frac{j}{|U_i| - r + j}\right) Q_{n-j+1}. \tag{4.25}$$

The following claim completes the proof of the theorem. QED.

Claim 4.2. For every $j \in \{2, \ldots, r\}$, we have

$$\left(1 - \frac{j}{|U_i| - r + j}\right) Q_{n-j+1} \geq Q_n$$

Proof. Let $j \in \{2, \ldots, r\}$. The given inequality is equivalent to $\frac{Q_{n-j+1}}{Q_n} \geq \frac{|U_i|-r+j}{|U_i|-r}$. Since U_i is the largest among $U_1, \ldots, U_t$ which together partition V, we have $|U_i| \geq \frac{n}{t}$. Consequently, $\frac{|U_i|-r+j}{|U_i|-r} = 1 + \frac{j}{|U_i|-r} \leq 1 + \frac{jt}{n-rt}$. Therefore, it suffices to prove that $\frac{Q_{n-j+1}}{Q_n} \geq 1 + \frac{jt}{n-tr}$. We case on the value of $n - j + 1$.

Case 1: Suppose that $n-j+1>rt$. Then, we have

$$\frac{Q_{n-j+1}}{Q_n}=\frac{\binom{n-t(r-2)}{2t}}{\binom{n-j+1-t(r-2)}{2t}}=\prod_{\ell=0}^{2t-1}\frac{n-t(r-2)-\ell}{n-j+1-t(r-2)-\ell}. \tag{4.26}$$

We consider two sub-cases based on the value of j.

Case 1.a: Suppose that $j>2t$. Then, we observe that

$$\begin{aligned}\prod_{\ell=0}^{2t-1}\frac{n-t(r-2)-\ell}{n-j+1-t(r-2)-\ell} &\geq \left(\frac{n-t(r-2)}{n-j+1-t(r-2)}\right)^{2t} &(4.27)\\ &=\left(1+\frac{j-1}{n-j+1-t(r-2)}\right)^{2t} &(4.28)\\ &\geq 1+\frac{2t(j-1)}{n-j+1-t(r-2)} &(4.29)\\ &\geq 1+\frac{jt+(j-2)t}{n-rt} &(4.30)\\ &\geq 1+\frac{jt}{n-rt}. &(4.31)\end{aligned}$$

We use $j>2t$ in the second to last inequality and $j\geq 2$ in the final inequality.

Case 1.b: Suppose that $j\leq 2t$. Then we can cancel additional terms from the right hand side of equation (4.26) to obtain that

$$\begin{aligned}\prod_{\ell=0}^{2t-1}\frac{n-t(r-2)-\ell}{n-j+1-t(r-2)-\ell} &=\prod_{\ell=0}^{j-2}\frac{n-t(r-2)-\ell}{n-tr-\ell} &(4.32)\\ &\geq\left(\frac{n-t(r-2)}{n-tr}\right)^{j-1} &(4.33)\\ &=\left(1+\frac{2t}{n-rt}\right)^{j-1} &(4.34)\\ &\geq 1+\frac{2t(j-1)}{n-rt} &(4.35)\\ &\geq 1+\frac{jt}{n-rt}. &(4.36)\end{aligned}$$

Thus, in either subcase, our desired inequality holds.

Case 2: Suppose that $n-j+1\leq rt$. Now the expression for Q_{n-j+1} is different. Since we still know that $n \geq rt+1$ and $j \leq r$, we conclude that

$$\begin{aligned}
\frac{Q_{n-j+1}}{Q_n} &= \frac{(rt+1)\binom{n-t(r-2)}{2t}}{2t+1} && (4.37)\\
&\geq \frac{(rt+1)\binom{rt+1-t(r-2)}{2t}}{2t+1} && (4.38)\\
&= rt+1 && (4.39)\\
&\geq 1+jt && (4.40)\\
&\geq 1+\frac{jt}{n-rt}. && (4.41)
\end{aligned}$$

Thus, our desired inequality holds in all cases. QED.

4.2.2 Finding Multiobjective Min-Cuts

In this section, we present Algorithm 4.2, which does *not* take a budget-vector as input, yet outputs any multiobjective min-cut (for any choice of budget-vector) with inverse polynomial probability. This is accomplished by returning a collection of cuts.

In contrast to Algorithm 4.1, all of the randomness in Algorithm 4.2 (except for the selection of a random cut in the base case) occurs upfront through the selection of a permutation of the hyperedges. Theorem 4.5 summarizes the guarantees of Algorithm 4.2.

Theorem 4.5. Let G be an r-rank, n-vertex hypergraph with t hyperedge-cost functions. Then, a fixed multiobjective min-cut F is in the collection returned by Algorithm 4.2 with probability

$$\frac{\Omega(n^{-2t})}{r2^{rt}}.$$

Moreover, the algorithm outputs at most n^{t-1} cuts.

Proof. We begin by showing the second part of the theorem. The algorithm outputs at most one cut for each choice of $n_1, \ldots, n_{t-1} \in [n]$ (or just one cut if $|V| \leq rt$). Hence, it outputs at most n^{t-1} cuts. We now argue that the algorithm retains the same success probability as Algorithm 4.1, for any fixed budget-vector b. Suppose $n \leq rt$. Then both Algorithm 4.2 and Algorithm 4.1 return $\delta(X)$ for a random subset X of the vertices of G. Thus, for any cut F, the two algorithms have the same probability of returning F. Henceforth, we assume $n > rt$.

We will view Algorithm 4.1 from a different perspective. In that algorithm, whenever we contract a hyperedge e, we choose, for some $i \in [t]$, a hyperedge according to the probability

```
MULTIOBJECTIVE-MIN-CUT($G, r, t, c_1, \ldots, c_t$):
  Input:  An $r$-rank hypergraph $G = (V, E)$ and
          hyperedge-cost functions $c_1, \ldots, c_t : E \to \mathbb{R}_{\geq 0}$.
  If $|V| \leq rt$:
      Pick a random subset $X$ of $V$ and return $\delta(X)$
  For $i = 1, \ldots, t$:
      $E_i \leftarrow \{e \in E \colon c_i(e) > 0\}$
      $\pi_i \leftarrow$ a permutation of $E_i$ generated by repeatedly choosing a not yet
          chosen hyperedge $e$ with probability proportional to $c_i(e)$
  $R \leftarrow \emptyset$
  For each sequence $n_1, \ldots n_t$ with $n \geq n_1 \geq n_2 \geq \cdots \geq n_{t-1} \geq n_t = rt$:
      $G' \leftarrow G$
      For $i = 1, \ldots, t$:
          While $|V(G')| > n_i$ and some hyperedge from $\pi_i$ is present in $G'$:
              $e \leftarrow$ the first hyperedge from $\pi_i$ that is still present in $G'$
              $G' \leftarrow G'/e$
      $X \leftarrow$ a random subset of $V(G')$
      Add $\delta(X)$ to $R$ if it is not already present
  Return $R$
```

Algorithm 4.2: MULTIOBJECTIVE MIN-CUT

distribution $\Pr[e = e'] = \frac{c_i(e')}{c_i(E)}$. In particular, the choice of i depends on which contractions have been made so far, but the choice of a particular hyperedge, given the choice of i, does not depend on our previous contractions, except for the fact that we do not contract hyperedges which have already been reduced to singletons. We note that allowing the contraction of singletons would not change the success probability of the algorithm. Therefore, we could modify Algorithm 4.1 so that it begins by selecting permutations $\pi_1, \ldots, \pi_t$ of $E_1, \ldots, E_t$ (where $E_i = \{e \in E \colon c_i(e) > 0\}$) as in Algorithm 4.2, and then whenever the algorithm asks to contract a random hyperedge with probability proportional to its weight under c_i, we instead contract the next hyperedge from π_i which is still present in the current hypergraph. This modification does not change at any step the probability that a particular hyperedge is the next contraction of a non-singleton hyperedge, and therefore the success probability of the algorithm remains exactly the same.

Viewing Algorithm 4.1 in this way, we note that when we reach the base case of $n \leq rt$, we will have contracted the first m_i hyperedges of each π_i, for some $m_1, \ldots, m_t \in \{0, \ldots |E|\}$. The crucial observation now is that interleaved contractions can be separated. That is, if we know m_i for every $i \in [t]$, the order in which we do the contractions is irrelevant: we will get the same resulting hypergraph if we contract the first m_1 hyperedges from π_1, then contract the first m_2 hyperedges from π_2, and so on up through the first m_t hyperedges from π_t instead of the interleaved contractions. Let n_1 be the number of vertices in the hypergraph obtained after contracting the first m_1 hyperedges from π_1, subsequently, let n_2 be the number of vertices in the hypergraph obtained after contracting the first m_2 hyperedges from π_2 and so on.

When we view Algorithm 4.1 in this way, it is only the choice of the values $n_1, \ldots, n_t$ that depends on the budgets, while the choice of the permutation π_i does not depend on the budgets. Algorithm 4.2 is running exactly the version of Algorithm 4.1 that we have just described, except that instead of choosing $n_1, \ldots, n_t$ based on the budgets, it simply tries all possible options (which will certainly include whichever $n_1, \ldots, n_t$ Algorithm 4.1 would use for the given input budget-vector). Therefore for every fixed choice of budget-vector b, and every fixed b-multiobjective min-cut F, the probability that F is in the collection R output by the algorithm is at least as large as the probability that F is the cut output by Algorithm 4.1. By Theorem 4.4, this probability is $\frac{\Omega(n^{-2t})}{r2^{rt}}$, as desired. QED.

We derive Theorem 4.1 from Theorem 4.5 now.

Theorem 4.1. The number of multiobjective min-cuts in an r-rank, n-vertex hypergraph with t hyperedge-cost functions is $O(r2^{rt}n^{3t-1})$.

Proof. Let x be the number of multiobjective min-cuts in the hypergraph. By Theorem 4.5, the expected number of multiobjective min-cuts output by our algorithm MULTIOBJECTIVE MIN-CUT (i.e., Algorithm 4.2) is $(x/r2^{rt})\Omega(n^{-2t})$. Theorem 4.5 also tells us that the algorithm outputs at most n^{t-1} cuts. Therefore, $x = r2^{rt} \cdot O(n^{3t-1})$. QED.

4.2.3 Enumerating Multiobjective Min-Cuts and Pareto-Optimal Cuts

In this section, we give algorithms to enumerate all multiobjective min-cuts and pareto-optimal cuts in polynomial time.

We first give a polynomial time algorithm to enumerate all multiobjective min-cuts. We execute our algorithm for MULTIOBJECTIVE MIN-CUT (i.e., Algorithm 4.2) a sufficiently large number of times so that it succeeds with high probability (i.e., with probability at least $1 - 1/n$): In particular, executing it $r^2t2^{rt}O(n^{2t}\log n)$ many times gives us a collection $\mathcal{C}$ that is a superset of the collection $\mathcal{C}_{MO}$ of multiobjective min-cuts with high probability. Moreover, the size of the collection $\mathcal{C}$ is $r^2t2^{rt}O(n^{3t-1}\log n)$. We can prune $\mathcal{C}$ to identify $\mathcal{C}_{MO}$ in polynomial-time as follows: remove every cut $F \in \mathcal{C}$ for which there exists a cut $F' \in \mathcal{C}$ with $c_t(F') < c_t(F)$ and $c_i(F') \le c_i(F)$ for every $1 \le i \le t-1$.

Next, we give a polynomial time algorithm to enumerate pareto-optimal cuts. By containment relation (4.1), it suffices to identify all pareto-optimal cuts in the collection $\mathcal{C}$. For this, we only need a polynomial-time procedure to verify if a given cut F is pareto-optimal. Algorithm 4.3 gives such a procedure. It essentially searches for a cut that dominates the given cut F by running our algorithm for b-Multiobjective Min-Cut with t different budget-vectors.

```
Verify-Pareto-Optimality($G, r, t, c_1, \ldots, c_t, F$):
  Input: An $r$-rank hypergraph $G = (V, E)$, cost functions $c_1, \ldots, c_t : E \to \mathbb{R}_{\geq 0}$,
         and a cut $F$ in $G$
  For $i = 1, \ldots, t$:
      $\vec{c} \leftarrow (c_1, \ldots c_{i-1}, c_{i+1}, \ldots, c_t, c_i)$
      $\vec{b} \leftarrow (c_1(F), \ldots, c_{i-1}(F), c_{i+1}(F), \ldots, c_t(F))$
      For $j = 1, \ldots, r2^{rt}O(n^{2t} \log n)$:
          $F' \leftarrow b$-Multiobjective-Min-Cut($G, r, t, \vec{c}, \vec{b}$)
          If $F'$ is a $b$-multiobjective cut in $(G, \vec{c})$ and $c_i(F') < c_i(F)$:
              Return FALSE
  Return TRUE
```

Algorithm 4.3: Verify pareto-optimality of a given cut

Theorem 4.6. Let G be an r-rank, n-vertex hypergraph G with t hyperedge-cost functions $c_1, \ldots, c_t : E(G) \to \mathbb{R}_{\geq 0}$ such that the hypergraph induced by hyperedges of positive c_i cost is connected for every $i \in [t]$, and let F be a cut in G. If F is a pareto-optimal cut, then Algorithm 4.3 returns TRUE, and if F is not a pareto-optimal cut, then the algorithm returns FALSE with high probability. Moreover, the algorithm can be implemented to run in polynomial time.

Proof. The run-time of the algorithm is polynomial since Algorithm 4.1 runs in polynomial time. Algorithm 4.3 returns false only if it finds a cut that dominates the input cut F. If F is pareto-optimal, no such cut will exist, and therefore the algorithm will return true.

Next, suppose the input cut F is not pareto-optimal. Then F must be dominated by some other cut F'. Let $i \in [t]$ be such that $c_i(F') < c_i(F)$ (such an i is guaranteed to exist by the definition of domination). Let b be a budget-vector of the costs of F under the cost functions other than c_i and let c' be c with c_i moved to the end of the vector of cost functions. Then F will not be a b-multiobjective min-cut in (G, c'), since F' also satisfies b, but has a lower c_i-cost. Therefore, any b-multiobjective min-cut will dominate F (since it will also satisfy b and cannot have higher c_i cost than F'). Thus, if any of our $r2^{rt}n^{2t} \log n$ calls to our algorithm for b-Multiobjective-Min-Cut (i.e., Algorithm 4.1) for this value of i returns a multiobjective min-cut, then the algorithm will return false. We recall that the hypergraph induced by hyperedges of positive c_i cost is connected. By Theorem 4.4, our algorithm for b-Multiobjective Min-Cut returns a b-multiobjective min-cut with probability $\frac{1}{r2^{rt}}\Omega(\frac{1}{n^{2t}})$.

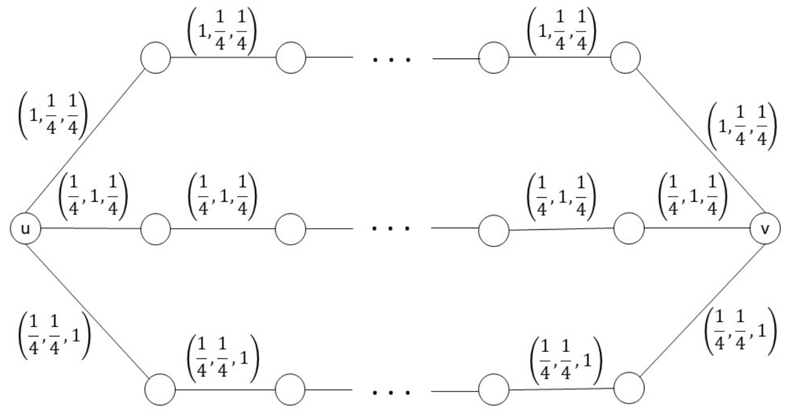
u
v
$\left(1,\frac{1}{4},\frac{1}{4}\right)$
$\left(\frac{1}{4},1,\frac{1}{4}\right)$
$\left(\frac{1}{4},\frac{1}{4},1\right)$
. . .

Figure 4.1: An illustration of our lower bound construction for $t = 3$.

Therefore, if we run this algorithm $r2^{rt}O(n^{2t}\log n)$ times, a b-multiobjective min-cut will be returned at least once with high probability, and our algorithm will correctly return false. QED.

4.2.4 Lower Bounds

In this section, we discuss lower bounds on the number of distinct pareto-optimal cuts in n-vertex hypergraphs. Karger gave a family of graphs with $n^{t/2}$ parametric min-cuts [82]. We recall that every parametric min-cut is a pareto-optimal cut by the containment relation (4.1). Thus, $n^{t/2}$ is also a lower bound on the number of pareto-optimal cuts in n-vertex hypergraphs. To the best of the authors' knowledge, this is the best lower bound on the number of pareto-optimal cuts that is known in the literature. We give an $\Omega(n^t)$ (for constant t) lower bound on the number of pareto-optimal cuts in a graph. Our lower bound construction is different from that of Karger.

Theorem 4.7. For all positive integers t and n such that $n \geq t+2$, there exists an n-vertex graph G with associated edge-cost functions $c_1, \ldots, c_t : E(G) \to \mathbb{R}_+$ such that G has at least $\left(\frac{n-2}{t}\right)^t$ distinct pareto-optimal cuts.

Proof. For fixed n and t, construct a graph G as follows. The graph G has two special vertices u and v. The rest of the vertices are used to form t distinct paths between u and v with each path consisting of at least $\lfloor\frac{n-2}{t}\rfloor + 1 > \frac{n-2}{t}$ distinct edges. We assign edge costs as follows: If e is an edge in the i'th path, then $c_i(e) = 1$, while $c_j(e) = 1/(t+1)$ for every $j \in [t] \setminus \{i\}$. See Figure 4.1 for an example.

We will show that any cut which contains exactly one edge from each path is pareto-optimal. The number of such cuts is at least $\left(\frac{n-2}{t}\right)^t$, since each path has at least $(n-2)/t$ edges, so this will suffice to prove the theorem.

We observe that any cut contains either exactly one edge from each path or at least two edges from some path. Any cut F which contains exactly one edge from each path will have $c_i(F) = 2t/(t+1)$ for every $i \in [t]$. Any cut F' that contains at least two edges from some path $i \in [t]$ will have $c_i(F') = 2 > 2t/(t+1)$. Therefore no cut which contains two edges from the same path can dominate a cut which contains exactly one edge from each path. Furthermore, if two different cuts each contain exactly one edge from all paths, then they both have the same cost under every cost function, and thus neither can dominate the other. We conclude that every cut which contains exactly one edge from each path is pareto-optimal. QED.

Remark 4.1 The lower bound of $\left(\frac{n-2}{t}\right)^t$ from Theorem 4.7 is still significantly smaller than the $O(n^{3t-1})$ upper bound from Theorem 4.1. We believe that this gap comes from the slack in the analysis of our randomized algorithms.

Remark 4.2 We note that the construction in Theorem 4.7 also shows that there exists a budget-vector $b \in \mathbb{R}_+^{t-1}$ such that the number of b-multiobjective min-cuts is $\Omega(n^t)$: consider budget values $b_i = (2t)/(t+1)$ for every $i \in [t-1]$. We emphasize that since not every multiobjective min-cut is pareto-optimal, this lower bound does not imply the one from Theorem 4.7. Since distinct pareto-optimal cuts need not be b-multiobjective min-cuts for the same vector b, the bound in Theorem 4.7 does not immediately imply this bound either.

4.2.5 Alternative Proof of a Strongly Polynomial Bound on the Number of Multiobjective Min-Cuts

In this section, we give an alternative proof due to Zenklusen showing that the number of multiobjective min-cuts in constant rank hypergraphs for constant many objective functions (i.e., when both r and t are $O(1)$) is strongly polynomial. The proof is constructive and leads to a deterministic algorithm to enumerate all multiobjective min-cuts/pareto-optimal cuts in strongly polynomial time.

Let $G = (V, E)$ be an r-rank, n-vertex hypergraph with t hyperedge-cost functions $c_1, \ldots, c_t : E \to \mathbb{R}_{\geq 0}$, such that the hypergraph induced by hyperedges of positive c_t cost is connected, where $r, t = O(1)$. For a subset $F \subseteq E$, we will write $c(F) = (c_1(F), \ldots, c_t(F))$ to denote the vector of cost function values. A cut F is *supported* if there exists $\mu_1, \ldots, \mu_t \geq 0$

such that any cut F' that is minimum with respect to the cost function given by $c_\mu(e) := \sum_{i=1}^{t} \mu_i c_i(e)$ for all $e \in E$ satisfies $c_i(F) = c_i(F')$ for every $i \in [t]$.

Let

$$P = \text{convex-hull}(\{(c_1(\delta(S)), \ldots, c_t(\delta(S))) : \emptyset \neq S \subsetneq V\}) + \mathbb{R}^t_{\geq 0} \subset \mathbb{R}^t \tag{4.42}$$

be the dominant (or up-closure) of the convex-hull of cost-vectors corresponding to cuts. Thus, P can equivalently be defined as the dominant of the convex-hull of all supported pareto-optimal cuts. Hence, the vertices of the polyhedron P are precisely the vectors $c(F)$, where F is a supported pareto-optimal cut.

We now give an algorithm to enumerate all multiobjective min-cuts/pareto-optimal cuts. We note that P is full-dimensional because it is up-closed and the graph G is connected with $|V| \geq 2$. Aissi, Mahjoub, McCormick, and Queyranne [81] showed that the number of supported pareto-optimal cuts is $\tilde{O}(n^{rt})$ and they can all be enumerated in time $\tilde{O}(n^{rt})$. Hence, P only has a strongly polynomial number of vertices, i.e., it has $\tilde{O}(n^{rt})$ many vertices. Consequently, the number of facets of P is $\tilde{O}(n^{rt^2})$ (since each facet is uniquely defined by $t-1$ linearly independent edge directions of the facet, each of which is either the difference of two vertices or one of the axis-parallel directions $e_1, \ldots, e_t$—stronger bounds can be obtained, for example, by using known bounds on the number of facets as a function of the number of vertices [139]). Let $\mathcal{F}$ be the set of facets of P. By the result of Aissi, Mahjoub, McCormick, and Queyranne, we have that $|\mathcal{F}| = \tilde{O}(n^{rt^2})$ and all these facets can be enumerated in $\tilde{O}(n^{rt^2})$ time. For each facet $F \in \mathcal{F}$, there is a parameteric cost c_F, which is a conic combination of the costs $c_1, \ldots, c_t$ such that F is the set of all points of P that minimize the linear objective c_F. Since the hypergraph induced by hyperedges of positive c_t cost is connected, no cut has zero cost with respect to c_t, and therefore, it follows that the origin 0 is not in P, and the minimum-cost cut with respect to any function c_F for $F \in \mathcal{F}$ has strictly positive value. The central idea behind the enumeration of all multiobjective min-cuts is the following result showing that every multiobjective min-cut is a t-approximate min-cut with respect to one of the cost functions in $\{c_F : F \in \mathcal{F}\}$.

Theorem 4.8. *Let $R \subseteq E$ be a multiobjective min-cut such that $c_t(R) > 0$. Then, there exists $F \in \mathcal{F}$ such that*

$$c_F(R) \leq t \min\{c_F(\delta(S)) : \emptyset \neq S \subsetneq V\}.$$

A strongly polynomial bound of $\tilde{O}(n^{rt^2})$ on the number of multiobjective min-cuts follows from Theorem 4.8, since for each $F \in \mathcal{F}$, there are $O(n^{t^2})$ many t-approximate min-cuts with respect to c_F (which follows by the results of Kogan and Krauthgamer [20]).

Proof of Theorem 4.8. Let R be a b-multiobjective min-cut for some $b \in \mathbb{R}^{t-1}$ and let $x := c(R) \in \mathbb{R}^t_{\geq 0}$. Consider the segment between x and the origin 0. Since the origin is not in P, the segment from x to 0 will leave P at some point. Let $\lambda > 0$ be the smallest value such that $\lambda x \in P$, and let $y = \lambda x$. Hence, y is the point where the segment from x to 0 is about to leave the polyhedron P. Let $F \in \mathcal{F}$ be a facet of P containing y and certifying that $\mu x \notin P$ for any $\mu < \lambda$. Hence, $\{\mu x : \mu \in \mathbb{R}\}$ crosses the facet F when going from $\mu > \lambda$ to $\mu < \lambda$. Formally, this is equivalent to choosing F to be a facet on which y lies and such that $c_F^T y > 0$.

Let Γ be the vertices of P, which correspond to cost-vectors of cuts. By definition of P, and the fact that $y \in P$, we can write y as a convex combination of vectors in Γ and a vector with non-negative entries. Formally,

$$y = \sum_{i=1}^{\ell} \beta_i z_i + \sum_{i=1}^{t} \gamma_i e_i, \tag{4.43}$$

where $z_i \in \Gamma$ and $\beta_i > 0$ for every $i \in [\ell]$, $\sum_{i=1}^{\ell} \beta_i = 1$, $\gamma_i \geq 0$ for every $i \in [t]$, and e_i is the all-zeroes vector with a single 1 in the i'th component. By Carathéodory's Theorem, we can choose $\ell \leq t$ since y is on a face of P of dimension at most $t-1$.

Let

$$j := \arg\max\{\beta_i : i \in [\ell]\}. \tag{4.44}$$

Since the β_is are non-negative and sum to 1, and $\ell \leq t$, we have

$$\beta_j \geq \frac{1}{t}. \tag{4.45}$$

Moreover, by expression (4.43), we have that $\beta_j z_j \leq y$. Hence,

$$\frac{1}{\lambda} \beta_j z_j \leq \frac{1}{\lambda} y = x. \tag{4.46}$$

Suppose $\beta_j/\lambda > 1$. Then $z_j(i) < (\beta_j/\lambda) z_j(i) \leq x(i)$ for every $i \in [t]$ satisfying $z_j(i) > 0$. Thus, the cut R' whose cost-vector is $z_j = c(R')$ is also a b-multiobjective min-cut with smaller objective value than R (i.e., $c_t(R') < c_t(R)$), a contradiction to optimality of R. Hence, $\beta_j/\lambda \leq 1$. Consequently,

$$\lambda \geq \beta_j \geq \frac{1}{t}, \tag{4.47}$$

and we get

$$c_F^T x = \frac{1}{\lambda} c_F^T y \leq t \cdot c_F^T y \leq t \cdot \min\{c_F(\delta(S)) : \emptyset \neq S \subsetneq V\}. \tag{4.48}$$

The last inequality above is because y lies on F and is therefore a minimizer of P along the direction c_F. QED.

4.3 NODE-BUDGETED MULTIOBJECTIVE MIN-CUTS

In this section, we give algorithms to find min-cuts that satisfy node-weighted budget constraints. Theorem 4.2 will be a consequence of these algorithms. We begin by formally defining the problem.

Let $G = (V, E)$ be a hypergraph with hyperedge-cost function $c : E \to \mathbb{R}_+$. Let $w_1, \ldots, w_t : V \to \mathbb{R}_{\geq 0}$ be vertex-weight functions. Let $c(F) = \sum_{e \in F} c(e)$ for $F \subseteq E$, and $w_i(U) = \sum_{v \in U} w_i(v)$ for $U \subseteq V$. The following definition will be useful in defining node-budgeted multiobjective min-cuts.

Definition 4.4. For a budget-vector $b \in \mathbb{R}^t_{\geq 0}$,

1. a vertex $v \in V$ is *feasible* if $w_i(v) \leq b_i$ for all $i \in [t]$ and *infeasible* otherwise and
2. a set of vertices $S \subseteq V$ is *feasible* if $w_i(S) = \sum_{v \in S} w_i(v) \leq b_i$ for all $i \in [t]$ and *infeasible* otherwise.

We recall the definition of node-budgeted multiobjective min-cuts.

Definition 4.5. For a budget-vector $b \in \mathbb{R}^t_{\geq 0}$, a set $F \subseteq E$ is a *node-budgeted b-multiobjective min-cut* if $F = \delta(X)$ for some feasible set $\emptyset \subsetneq X \subsetneq V$, and $c(F)$ is minimum among all such subsets of E. A set $F \subseteq E$ is a *node-budgeted multiobjective min-cut* if there exists a budget-vector $b \in \mathbb{R}^t_{\geq 0}$ for which F is a node-budgeted b-multiobjective min-cut.

The following will be a central problem of interest in this section.

Node-Budgeted b-Multiobjective Min-Cut

Given: A hypergraph $G = (V, E)$ with vertex-weight functions $w_1, \ldots, w_t \colon V \to \mathbb{R}_{\geq 0}$, a hyperedge-cost function $c \colon E \to \mathbb{R}_+$, and a budget-vector $b \in \mathbb{R}^t_{\geq 0}$.

Goal: A node-budgeted b-multiobjective min-cut.

Problem 4.2: Node-Budgeted b-Multiobjective Min-Cut

4.3.1 Constant-Rank Hypergraphs

In this section, we design an algorithm to find node-budgeted b-multiobjective min-cuts in constant-rank hypergraphs in polynomial-time and then prove the first part of Theorem 4.2. We use Algorithm 4.4 to solve node-budgeted b-multiobjective min-cuts in constant-rank hypergraphs. It essentially runs the standard random contraction algorithm for min-cut with an additional step that deterministically contracts all infeasible vertices together. We summarize the guarantees of this algorithm in Theorem 4.9. We will subsequently use Theorem 4.9 to prove the first part of Theorem 4.2.

```
NODE-BUDGETED-b-MULTIOBJECTIVE-MIN-CUT-CONSTANT-RANK($G, r, t, w, c, b$):
  Input:  An $r$-rank hypergraph $G = (V, E)$, a positive integer $t$,
          a vector $w$ of vertex-weight functions with $w_i : V \to \mathbb{R}_+$ for $i \in [t]$,
          a cost function $c : E \to \mathbb{R}_+$, and budget-vector $b \in \mathbb{R}^t_+$
  Contract all infeasible vertices of $G$ into a single vertex
  If $|V| \le r + 1$:
      $X \leftarrow$ a random subset of $V$
      Return $\delta(X)$
  $e \leftarrow$ a random hyperedge chosen according to $\Pr[e = e'] = \frac{c(e')}{c(E)}$
  $(G, w) \leftarrow (G, w)/e$
  Return NODE-BUDGETED-b-MULTIOBJECTIVE-MIN-CUT-CONSTANT-RANK($G, r, t, w, c, b$)
```

Algorithm 4.4: NODE-BUDGETED b-MULTIOBJECTIVE MIN-CUT in constant-rank hypergraphs

Theorem 4.9. Let $G = (V, E)$ be an r-rank n-vertex hypergraph with hyperedge-cost function $c : E \to \mathbb{R}_+$ and vertex-weight functions $w_1, \dots, w_t : V \to \mathbb{R}_{\ge 0}$ and a budget-vector $b \in \mathbb{R}^t_{\ge 0}$. Let F be an arbitrary node-budgeted b-multiobjective min-cut in G. Then Algorithm 4.4 returns F with probability at least $\frac{1}{2^{r+1}\binom{n}{2}}$. Moreover, the algorithm can be implemented to run in polynomial time.

Proof. We first analyze the run-time. Each recursive call reduces the number of vertices, so the total number of recursive calls is at most n. Apart from the recursion, the algorithm only performs contractions and returns a random cut, all of which can be done in polynomial time.

Now we analyze the success probability. Let $Q_n := \frac{1}{2^{r+1}}\binom{n}{2}^{-1}$. We will show that the algorithm returns F with probability at least Q_n. We prove this by induction on n. Let $\emptyset \subsetneq X \subsetneq V$ be a feasible set with $\delta(X) = F$. We first note that all vertices in X must be feasible. Therefore, the cut X cannot be destroyed when all infeasible vertices are contracted together. This means that if G has multiple infeasible vertices, they will simply be contracted to yield a smaller hypergraph with at most one infeasible vertex where F is still a node-budgeted b-multiobjective min-cut. Therefore, we will assume without loss of generality that G contains at most one infeasible vertex.

For the base case, we consider $n \le r+1$. In this case, the algorithm returns $\delta(X)$ for a random $X \subseteq V$. There are 2^n possible choices for X, and F for at least one of them. Thus, the probability that the algorithm returns F is at least $\frac{1}{2^n} \ge \frac{1}{2^{r+1}} \ge Q_n$.

We now prove the induction step. Let $n > r+1$ and assume that the theorem holds for all hypergraphs with at most $n-1$ vertices and rank at most r. We begin by showing the following claim.

Claim 4.3. The algorithm outputs F with probability at least the optimum value of the following linear program.

$$\begin{aligned} \underset{x_2,\dots,x_r,y_2,\dots,y_r}{\text{minimize}} \quad & \sum_{j=2}^{r}(x_j - y_j)Q_{n-j+1} \\ \text{subject to} \quad & 0 \le y_j \le x_j \ \forall j \in \{2,\dots,r\} \\ & \sum_{j=2}^{r} x_j = 1 \\ & (n-1)\sum_{j=2}^{r} y_j \le \sum_{j=2}^{r} j \cdot x_j \end{aligned}$$

Proof. Since $n > r+1$, the algorithm will contract a randomly chosen hyperedge and recurse. Let e' be the random hyperedge chosen by the algorithm. If $e' \notin F$, then F will still be a node-budgeted b-multiobjective min-cut in G/e'. We observe that G/e' is a hypergraph with $n - |e'| + 1$ vertices and that the rank of G/e is at most the rank of G. Therefore, if $e' \notin F$, the algorithm will output F with probability at least $Q_{n-|e'|+1}$

Let

$$E_j := \{e \in E \colon |e| = j\}, \tag{4.49}$$

$$x_j := \Pr[e' \in E_j] = \frac{c(E_j)}{c(E)}, \text{ and} \tag{4.50}$$

$$y_j := \Pr[e' \in E_j \cap F] = \frac{c(E_j \cap F)}{c(E)}. \tag{4.51}$$

We note that E_j is the set of hyperedges of size j, x_j is the probability of picking a hyperedge of size j to contract, and y_j is the probability of picking a hyperedge of size j from F to contract. We know that

$$\Pr[\text{Algorithm returns the cut } F] \geq \sum_{j=2}^{r}(x_j - y_j)Q_{n-j+1} \tag{4.52}$$

The values of x_j and y_j will depend on the structure of G. Nevertheless, we can deduce some relationships between them. Since $0 \leq c(E_j \cap F) \leq c(E_j)$ for every $j \in \{2, \ldots, r\}$, we know that

$$0 \leq y_j \leq x_j \text{ for every } j \in \{2, \ldots, r\}. \tag{4.53}$$

Moreover, x_j is the probability of picking a hyperedge of size j. Hence,

$$\sum_{j=2}^{r} x_j = 1. \tag{4.54}$$

By the definition of a node-budgeted b-multiobjective min-cut, we have that $c(F) \leq c(\delta(X))$ for every feasible set X with $\emptyset \neq X \subsetneq V$. In particular, for every feasible vertex v, we have that $c(F) \leq c(\delta(v))$. Since we have assumed that G has at most 1 infeasible vertex, it has at least $n-1$ feasible vertices, and thus,

$$c(F) \leq \frac{\sum_{v:v \text{ is feasible}} c(\delta(v))}{|\{v\colon v \text{ is feasible}\}|} \leq \frac{\sum_{v\in V} c(\delta(v))}{n-1} = \frac{\sum_{e\in E}|e|c(e)}{n-1} = \frac{\sum_{j=2}^{r} j \cdot c(E_j)}{n-1}. \tag{4.55}$$

Thus we have that,

$$\sum_{j=2}^{r} y_j = \Pr[e' \in F] = \frac{c(F)}{c(E)} \leq \frac{1}{n-1}\sum_{j=2}^{r} j \cdot x_j. \tag{4.56}$$

The minimum value of our lower bound in equation (4.52) over all choices of x_j and y_j that satisfy inequalities (4.53), (4.54), and (4.56) is a lower bound on the probability that the algorithm outputs F. QED.

Claim 4.3 tells us that the algorithm outputs F with probability at least the optimum value of the linear program from the claim. This linear program is exactly the linear program from Lemma 4.1 with $\gamma = n-1$ and $f(n) := Q_n$. Since $n > r+1$, we have that $n \geq n-1 \geq r+1$. Therefore, we can apply Lemma 4.1 to conclude that

$$\Pr[\text{Algorithm returns the cut } F] \geq \min_{j\in\{2,\ldots,r\}} \left(\left(1 - \frac{j}{n-1-r+j}\right) Q_{n-j+1}\right). \tag{4.57}$$

It remains to show that $\min_{j\in\{2,\dots,r\}}((1-\frac{j}{n-1-r+j})Q_{n-j+1}) \geq Q_n$. Let $j \in \{2,\dots,r\}$. Then it suffices to show that $\frac{Q_{n-j+1}}{Q_n} \geq \frac{n-1-r+j}{n-1-r} = 1 + \frac{j}{n-1-r}$. We have

$$\frac{Q_{n-j+1}}{Q_n} = \frac{\binom{n}{2}}{\binom{n-j+1}{2}} = \frac{n(n-1)}{(n-j+1)(n-j)} \geq \left(\frac{n}{n-j+1}\right)^2 = \left(1+\frac{j-1}{n-j+1}\right)^2, \quad (4.58)$$

and

$$\left(1+\frac{j-1}{n-j+1}\right)^2 \geq 1+\frac{2(j-1)}{n-j+1} \geq 1+\frac{j}{n-j+1} \geq 1+\frac{j}{n-1-r}. \quad (4.59)$$

The last two inequalities follow from the fact that $2 \leq j \leq r$. QED.

We now restate and prove the first part of Theorem 4.2. At a high level, our approach will be similar to the proof of Theorem 4.1. We will modify the algorithm for Node-Budgeted b-Multiobjective Min-Cut to obtain Algorithm 4.5 which outputs a collection $\mathcal{C}$ of $r \cdot O(n^t)$-many cuts such that every node-budgeted multiobjective min-cut is in $\mathcal{C}$ with probability $\frac{1}{2^r} \cdot \Omega(\frac{1}{n^2})$. The analysis has a few subtleties that distinguish it from the edge-budgeted version, so we include the full details.

```
Node-Budgeted-Multiobjective-Min-Cut-Constant-Rank($G, r, t, w, c$):
  Input:  An $r$-rank hypergraph $G = (V, E)$, a positive integer $t$,
          a vector $w$ of vertex-weight functions with $w_i : V \to \mathbb{R}_{\geq 0}$ for $i \in [t]$,
          and a cost function $c : E \to \mathbb{R}_+$
   $\pi \leftarrow$ a permutation of $E$ generated by repeatedly choosing a not yet
      chosen hyperedge with probability proportional to $c(e)$
  $R \leftarrow \emptyset$
  $T \leftarrow \emptyset$
  For $n' = 2, \dots, n$:
      $G' \leftarrow G$
      Contract hyperedges from $G'$ in the order given by $\pi$ until $G'$ has at most $n'$ vertices
      For each $x_1, \dots, x_t$ such that $x_i = c_i(v)$ for some $v \in G'$ for all $i \in [t]$:
          $G'' \leftarrow G'$
          Contract together all vertices $v$ in $G''$ which have $c_i(v) > x_i$ for some $i$
          If $r + 2 > |V(G'')| > 1$ and $V(G'') \notin T$:
              $S \leftarrow$ a random subset of $V(G')$ with $\emptyset \subset S \subset V(G')$
              Add $V(G'')$ to $T$
              Add $\delta(S)$ to $R$ if it is not already present
  Return $R$
```

Algorithm 4.5: Node-Budgeted Multiobjective Min-Cut in constant-rank hypergraphs

Theorem 4.10. The number of node-budgeted multiobjective min-cuts in an r-rank, n-vertex hypergraph with t vertex-weight functions is $O(r2^r n^{t+1})$.

Proof. Let $G = (V, E)$ be an r-rank n-vertex hypergraph with vertex-weight functions $w_1, \dots, w_t : V \to \mathbb{R}_{\geq 0}$. We will denote $w = (w_1, \dots, w_t)$ and the hypergraph with the vertex-weight functions by the tuple (G, w) for conciseness. We first show that for any cut $F \subseteq E$ which is a node-budgeted b-multiobjective min-cut in (G, w) for some budget-vector $b \in \mathbb{R}^t_{\geq 0}$, the cut F is among the cuts returned by Algorithm 4.5 with probability $\Omega(2^{-r}n^{-2})$.

We will view Algorithm 4.4 from a different perspective. That algorithm alternates between contracting together infeasible vertices and contracting random hyperedges until the hypergraph has at most $r+1$ vertices. We note that the probability that a given hyperedge e is the next one contracted depends only on the cost of e relative to the other hyperedges. In particular it does not depend on which infeasible vertices have been contracted together. Therefore, we could modify Algorithm 4.4 so that it contracts random hyperedges until the hypergraph resulting from contracting all infeasible vertices together has at most $r+1$ vertices, at which point, it contracts all infeasible vertices together and returns a random cut in the resulting hypergraph. This modified version of the algorithm would retain the same success probability as the original version. In this modified algorithm, the next contraction does not depend at all on the previous contractions, so we can choose a uniform random permutation of the hyperedges at the start of the algorithm and simply contract hyperedges from that permutation until we can contract all infeasible vertices to obtain a hypergraph containing at most $r+1$ vertices.

Let U be the set of all feasible vertices in G, and for each $i \in [t]$, let $x_i = \max_{u \in U} w_i(u)$. Since all vertices in U are feasible, we know that for every $u \in U$, we have $w_i(u) \leq x_i \leq b_i$ for every $i \in [t]$. Now consider an infeasible vertex v in G. Since v violates the budget-vector b, there must be some $i \in [t]$ such that $w_i(v) > b_i \geq x_i$. Therefore, if we wish to contract together all infeasible vertices in G, it suffices to find, for each $i \in [t]$, the feasible vertex u_i with maximum weight under w_i, and then contract together all vertices whose w_i weight is greater than that of u_i for some $i \in [t]$. In particular, we can further modify our modified version of Algorithm 4.4 to use this method of contracting all infeasible vertices, and the success probability will still remain $\Omega(2^{-r}n^{-2})$.

Algorithm 4.5 is running exactly the version of the algorithm that we have just described, with two additional modifications: (1) Instead of contracting hyperedges from π until the contraction of infeasible vertices would yield a hypergraph with at most $r+1$ vertices, it simply tries all possible stopping points for the contraction of hyperedges from π, and (2) instead of choosing the values $x_1, \dots, x_t$ based on a budget-vector b, it simply tries all possible values for $x_1, \ldots, x_t$. This means that, for any budget-vector b, Algorithm 4.5 will try the values of n' and $x_1, \ldots, x_t$ that the modified version of Algorithm 4.4 would use.

Therefore, by Theorem 4.9, we know that for any fixed budget-vector b and every fixed node-budgeted b-multiobjective min-cut F, the probability that F is among the cuts output by the algorithm is $\Omega(2^{-r}n^{-2})$.

Now we bound the number of cuts returned by the algorithm. We note that the algorithm only adds a new cut to R if the set of vertices that the algorithm ends up with after performing all contractions has size between 2 and $r+1$ and is different from every set the algorithm has already obtained from previous combinations of parameters. We will show that, for a fixed G and π, the number of distinct sets of size between 2 and $r+1$ that we can obtain by contracting vertices in the way specified by the algorithm is at most rn^t.

There are at most n ways to choose the value of n', and also at most n choices for the values of $x_1, \ldots, x_{t-1}$. For fixed values of n' and $x_1, \ldots, x_{t-1}$, the choice of x_t determines the final set of vertices after contraction. Decreasing x_t can cause more vertices to become contracted (because some new vertex v may now have $w_t(v) > x_t$), but it cannot cause any vertex that was previously being contracted to no longer be contracted. Thus, there are most r distinct sets of vertices of size between 2 and $r+1$ that we could obtain by varying the value of x_t. Therefore the total number of distinct sets of size between 2 and $r+1$ that could result from contracting vertices in the way described in the algorithm is at most rn^t.

To finish the proof, let N be the number of node-budgeted multiobjective min-cuts in G. We have shown that our algorithm outputs $\Omega(\frac{N}{2^r n^2})$ of these cuts in expectation. But since our algorithm outputs at most rn^t cuts, we conclude that the number N of multiobjective min-cuts must be $O(r2^r n^{t+2})$. QED.

4.3.2 Arbitrary-Rank Hypergraphs

In this section, we present a polynomial-time algorithm for node-budgeted b-multiobjective min-cut in arbitrary-rank hypergraphs. The second part of Theorem 4.2 will follow from the correctness analysis of this algorithm.

We recall that global min-cut (without node-budgets) in arbitrary-rank hypergraphs already requires the *non-uniform* random contraction technique. We extend the non-uniform contraction technique of [138] for the node-budgeted variant. In addition, our algorithm will use the non-uniform contraction algorithm for global min-cut by [138] as a subroutine. We reproduce their algorithm for completeness in Algorithm 4.6 and state its guarantee in Theorem 4.11.

```
HYPERGRAPH-MIN-CUT($G, c$):
  Input:  A hypergraph $G = (V, E)$, and a cost function $c : E \to \mathbb{R}_+$
  Compute $\beta_e := \frac{|V|-|e|}{|V|} \cdot c(e)$ for every hyperedge $e \in E$
  If $\beta_e = 0$ for every hyperedge $e \in E(G)$, then return $E(G)$
  $e \leftarrow$ a random hyperedge of $G$ chosen with probability proportional to $\beta_e$
  Return HYPERGRAPH-MIN-CUT($G/e, c$)
```

Algorithm 4.6: HYPERGRAPH MIN-CUT

Theorem 4.11. [138] Algorithm 4.6 runs in polynomial time and returns any fixed min-cut of an n-vertex hypergraph G with hyperedge-cost function c with probability at least $\binom{n}{2}^{-1}$.

Now we describe our algorithm for solving node-budgeted b-multiobjective min-cut in arbitrary-rank hypergraphs. We recall that a vertex v is feasible if $w_i(v) \le b_i$ for all $i \in [t]$. Let U be the set of all feasible vertices in G. We emphasize that U is the set of all feasible vertices, but U may not be a feasible set—see Definition 4.4. Our algorithm chooses a hyperedge e to contract with probability proportional to

$$\alpha_e := \left(\frac{|U| - |e \cap U|}{|U|}\right) \cdot c(e) = \left(\frac{|U \setminus e|}{|U|}\right) \cdot c(e). \tag{4.60}$$

and recurses on the contracted graph. Our algorithm performs an additional step of contracting all infeasible vertices after each contraction step. The description of our algorithm is presented in Algorithm 4.7. We summarize the correctness probability and the run-time of Algorithm 4.7 in Theorem 4.12.

Theorem 4.12. Let $G = (V, E)$ be an n-vertex hypergraph, for some $n \ge 2$ with vertex-weight functions $w_1, \ldots, w_t : V \to \mathbb{R}_{\ge 0}$, cost function c: $E \to \mathbb{R}_+$ and budget-vector $b \in \mathbb{R}^t_{\ge 0}$. Then Algorithm 4.7 outputs a fixed node-budgeted b-multiobjective min-cut in G with probability at least

$$Q_n := \begin{cases} 1 & \text{if } n = 2, \\ \frac{1}{3}\binom{n-1}{2}^{-1} & \text{if } n \ge 3. \end{cases}$$

Moreover, the algorithm can be implemented to run in polynomial time.

Proof. We first analyze the run-time. Each recursive call in the algorithm reduces the number of vertices, so the total number of recursive calls is at most n. Apart from the recursion, the algorithm, verifies the feasibility of each vertex and of U, computes α_e for each hyperedge, and either performs a contraction or calls the algorithm for the ordinary hypergraph min-cut problem. All of these can be done in polynomial time.

```
NODE-BUDGETED-$b$-MULTIOBJECTIVE-MIN-CUT-ARBITRARY-RANK$(G, t, w, c, b)$:
  Input:  A hypergraph $G = (V, E)$, a positive integer $t$,
          a vector $w$ of vertex-weight functions with $w_i : V \to \mathbb{R}_{\geq 0}$ for $i \in [t]$,
          a cost function $c : E \to \mathbb{R}_+$, and a budget-vector $b \in \mathbb{R}^t_{\geq 0}$
  $U \leftarrow \{v \in V : v$ is feasible$\}$
  If $U = \emptyset$:
      Return INFEASIBLE
  Compute $\alpha_e := \frac{|U \setminus e|}{|U|} \cdot c(e)$ for every hyperedge $e \in E$
  If $\alpha_e = 0$ for every hyperedge $e \in E$:
      If $U$ is feasible:
          Return $\delta(U)$
      Return $E$
  Contract together all infeasible vertices in $G$
  If $U$ is feasible:
      Return HYPERGRAPH-MIN-CUT$(G, c)$
  $e \leftarrow$ a random hyperedge of $G$ chosen with probability proportional to $\alpha_e$
  Return NODE-BUDGETED-$b$-MULTIOBJECTIVE-MIN-CUT-ARBITRARY-RANK$(G, t, w, c, b)$
```

Algorithm 4.7: NODE-BUDGETED b-MULTIOBJECTIVE MIN-CUT in arbitrary-rank hypergraphs.

Now we analyze the correctness probability. Let $\mathcal{G}_n$ be the set of all tuples (G, t, w, c, b) where G is an n-vertex hypergraph, $t \in \mathbb{Z}_+$, $w_1, \ldots, w_t : V(G) \to \mathbb{R}_{\geq 0}$, $c : E(G) \to \mathbb{R}_+$, and $b \in \mathbb{R}^t_{\geq 0}$. That is, $\mathcal{G}_n$ is the set of all valid inputs to Algorithm 4.7 where the hypergraph has n vertices. For an input tuple T in the form just described, let $M(T)$ be the collection of b-multiobjective min-cuts for the input instance. Define

$$q_n := \min_{T \in \mathcal{G}_n} \min_{F \in M(T)} \Pr[\text{Algorithm returns } F \text{ on input } T]. \tag{4.61}$$

We will show that $q_n \geq Q_n$ for all $n \geq 2$. We proceed by induction on n. As a base case, when $n = 2$, we have $\alpha_e = 0$ for every e, and the algorithm outputs the unique cut with probability 1, so $q_2 = 1 = Q_2$.

We now show the induction step. Let G be a hypergraph on $n \geq 3$ vertices with associated costs, weights, and budgets, and let F be a b-multiobjective min-cut in G. Assume that $q_{n'} \geq Q_{n'}$ for $2 \leq n' < n$. We will show that the algorithm returns F with probability at least $Q_n = \frac{1}{3}\binom{n-1}{2}^{-1}$.

Suppose $\alpha_e = 0$ for every $e \in E$. This means that every hyperedge contains all of the feasible vertices. Let $\emptyset \subsetneq X \subsetneq V$ be a feasible set (one which does not violate the budgets). Then, every vertex in X must be feasible. Since every hyperedge contains all feasible vertices, $\delta(X)$ will either be all of the hyperedges (if X does not contain all feasible vertices) or all hyperedges which contain infeasible vertices (if X contains all feasible vertices). The latter is cheaper, so if the set U of all feasible vertices is still feasible, then we must have $F = \delta(U)$, and the algorithm always returns F. Otherwise, every feasible cut contains all hyperedges, so $F = E$ and again the algorithm always returns F. We hereafter assume that $\alpha_e > 0$ for some hyperedge e.

We note that if G has multiple infeasible vertices, the algorithm will contract them together to yield a hypergraph G' with $n' < n$ vertices and only one infeasible vertex. The probability that the algorithm returns F on input G will be the same as the probability that the algorithm returns F on input G'. From our induction hypothesis we know that this probability is at least $Q_{n'} \geq Q_n$. We hereafter assume that G has at most one infeasible vertex.

Next we consider the case where the set U is feasible. (We emphasize that although $w_i(v) \leq b_i$ for every $v \in U$, $i \in [t]$, by the definition of feasible vertices, it need not be the case that $\sum_{v \in U} w_i(v) \leq b_i$ for every $i \in [t]$. So this case does not occur always.) Since our vertex weights are all non-negative, if U is feasible, then every subset of U must be feasible as well. Any cut can be written as $\delta(X) = \delta(\overline{X})$ for some $X \subseteq V$. Since G has at most one infeasible vertex, either X or $\overline{X}$ must be a subset of U. This means that every cut in G must be feasible. Thus, in this case, the budgets are irrelevant and finding a node-budgeted b-multiobjective min-cut is the same as just finding an ordinary minimum cut with respect to the cost function c. In particular, this means that F is not only a node-budgeted b-multiobjective min-cut in G, but it is also a regular min-cut as well. Therefore by Theorem 4.11, the algorithm for Hypergraph Min-Cut (i.e., Algorithm 4.6) outputs F with probability at least $\binom{n}{2}^{-1}$. Consequently, Algorithm 4.7 outputs F with probability at least $\binom{n}{2}^{-1}$. Since $n \geq 3$, we have that $\binom{n}{2}^{-1} \geq \frac{1}{3}\binom{n-1}{2}^{-1} = Q_n$.

Finally, suppose that G has at most one infeasible vertex and that U is not feasible in G. Then, the algorithm contracts a hyperedge with probability proportional to α_e. Let e' be a random variable for the contracted hyperedge. Using the induction hypothesis, we obtain that

$$\Pr[\text{Alg. 4.7 returns } F \text{ on input } G] = \sum_{e \in E \setminus F} \Pr[e' = e] \cdot \Pr[\text{Alg. 4.7 returns } F \text{ on input } G/e] \tag{4.62}$$

$$\geq \sum_{e \in E \setminus F} \frac{\alpha_e}{\sum_{f \in E} \alpha_f} \cdot q_{n-|e|+1} \tag{4.63}$$

$$\geq \frac{1}{\sum_{e \in E} \alpha_e} \sum_{e \in E \setminus F} \alpha_e Q_{n-|e|+1}. \tag{4.64}$$

Now, Claims 4.4 and 4.5 complete the proof of the theorem. QED.

Claim 4.4. For every hyperedge $e \in E \setminus F$, we have

$$\alpha_e Q_{n-|e|+1} \geq c(e) Q_n.$$

Proof. Suppose $|e| = n-1$. Then $Q_{n-|e|+1} = 1$. Since U is not feasible, we know that F must contain every hyperedge that spans U. Since $e \in E \setminus F$, it follows that $|U \setminus e| > 0$. Therefore, $\alpha_e \geq \frac{c(e)}{n}$. We conclude that $\alpha_e Q_{n-|e|+1} = \alpha_e \geq \frac{c(e)}{n} \geq \frac{c(e)}{3}\binom{n-1}{2}^{-1} = c(e)Q_n$.

Next, suppose $|e| < n-1$. Then $Q_{n-|e|+1} = \frac{1}{3}\binom{n-|e|}{2}^{-1}$, and we have

$$\begin{aligned}
\alpha_e Q_{n-|e|+1} &= \frac{|U \setminus e| c(e)}{|U|} \cdot \frac{1}{3}\binom{n-|e|}{2}^{-1} && (4.65)\\
&\geq \frac{|U| - |e|}{|U|} \cdot \frac{1}{3}\binom{n-|e|}{2}^{-1} \cdot c(e) && (4.66)\\
&\geq \frac{n-1-|e|}{n-1} \cdot \frac{2}{3(n-|e|)(n-|e|-1)} \cdot c(e) \qquad (\text{since } |U| \geq n-1) && (4.67)\\
&\geq \frac{2}{3(n-1)(n-|e|)} \cdot c(e) && (4.68)\\
&\geq \frac{2}{3(n-1)(n-2)} \cdot c(e) && (4.69)\\
&= c(e) Q_n. && (4.70)
\end{aligned}$$

QED.

Claim 4.5.

$$\frac{c(E \setminus F)}{\sum_{e \in E} \alpha_e} \geq 1.$$

Proof. We consider the cut induced by a uniformly random feasible vertex. A hyperedge e belongs to such a cut with probability $\frac{|U \cap e|}{|U|} = 1 - \frac{|U \setminus e|}{|U|}$. Thus, the expected value of such a cut is $\sum_{e \in E}(1 - \frac{|U \setminus e|}{|U|})c(e) = c(E) - \sum_{e \in E} \alpha_e$. Since the value of the cut induced by a random feasible vertex is an upper bound on the value of a node-budgeted b-multiobjective min-cut, this means that $c(F) \leq c(E) - \sum_{e \in E} \alpha_e$. Rewriting this inequality gives $\sum_{e \in E} \alpha_e \leq c(E) - c(F) = c(E \setminus F)$, and the desired inequality follows. QED.

4.4 SIZE-CONSTRAINED MIN-k-CUT IN ARBITRARY-RANK HYPERGRAPHS

In this section, we consider the problem of finding a minimum cost k-cut subject to constant lower bounds on the weights of the partition classes and prove Theorem 4.3. Throughout this section, we assume that k is a constant. We focus on the cardinality case (i.e., unit-cost variant) for the sake of simplicity of exposition and mention that our algorithm also extends to arbitrary non-negative hyperedge costs.

We begin by formally defining the terminology. Let $G = (V, E)$ be a hypergraph. For a weight function $w : V \to \mathbb{Z}_+$, we call (G, w) a *vertex-weighted hypergraph.* We now define our main object of study, namely size-constrained minimum cuts.

Definition 4.6. Let $G = (V, E)$ be a hypergraph, $w : V \to \mathbb{Z}_+$ be a vertex-weight function, $k \geq 2$ be an integer, and $s \in \mathbb{Z}_+^k$ be a vector. A k-partition X of V is an *s-size-constrained k-partition* if $w(X_i) \geq s_i$ for every $i \in [k]$. A set of hyperedges $F \subseteq E$ is an *s-size-constrained k-cut* if $F = \delta(X)$ for some s-size-constrained k-partition X. An s-size-constrained k-cut of minimum cardinality is said to be an *s-size-constrained min-k-cut.* We will refer to s as the *size-constraint vector.*

The following is the central problem of interest in this section.

s-Size-Constrained Min-k-Cut
Given: A vertex-weighted hypergraph (G, w), a positive integer k, and a size-constraint vector $s \in \mathbb{Z}_+^k$.
Goal: An s-size-constrained min-k-cut.

Problem 4.3: s-Size-Constrained Min-k-Cut

We give a random contraction based algorithm for this problem. Given a hypergraph $G = (V, E)$ and a size-constraint vector $s \in \mathbb{Z}_+^k$, let $n = |V|$. We define $\sigma_j := \sum_{i=1}^{j} s_i$, and

$$\alpha_e := \frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}}. \tag{4.71}$$

With these definitions, we solve s-size-constrained min-k-cut using Algorithm 4.8. We prove Theorem 4.3 using this algorithm.

Theorem 4.3. Let $k \geq 2$ be a fixed positive integer and let $1 \leq s_1 \leq s_2 \leq \ldots \leq s_k$ be fixed positive integers. Let $G = (V, E)$ be an n-vertex hypergraph with hyperedge-cost function $c : E \to \mathbb{R}_+$. Then, there exists a polynomial-time algorithm that takes (G, c) as input and returns a fixed w-weighted s-size-constrained min-k-cut for any choice of vertex-weight function $w : V \to \mathbb{Z}_+$ with probability

$$\Omega\left(\frac{1}{n^{2\sigma_{k-1}+1}}\right),$$

```
s-Size-Constrained-Min-k-Cut(G, k, s):
    Input:  An n-vertex hypergraph $G = (V, E)$, an integer $k \geq 2$
            and size-constraint vector $s = (s_1, \ldots, s_k) \in \mathbb{Z}_+^k$
    If $n \leq \max\{2\sigma_{k-1}, \sigma_k\}$
        Pick a random k-partition $X$ of $V$ and return $\delta(X)$
    $S \leftarrow$ a random subset of $V$ of size $2\sigma_{k-1}$
    $X_i \leftarrow \emptyset$ for $i \in \{1, \ldots, k\}$
    For each $v \in S$:
        Pick a random integer $i \in \{1, \ldots, k\}$ and add $v$ to $X_i$
    $X_k \leftarrow X_k \cup (V \setminus S)$
    $X \leftarrow (X_1, \ldots, X_k)$
    If $X$ is a k-partition of $V$:
        $R \leftarrow \delta(X)$
    Else:
        $R \leftarrow E$
    Compute $\alpha_e := \frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}}$ for every $e \in E$
    If $\alpha_e = 0$ for every hyperedge $e \in E(G)$:
        Return $R$
    $e \leftarrow$ a random hyperedge of $G$ chosen with probability proportional to $\alpha_e$
    $R' \leftarrow$ s-Size-Constrained-Min-k-Cut$(G/e, k, s)$
    Return $R$ with probability $\frac{1}{n}$ and $R'$ with probability $\frac{n-1}{n}$
```

Algorithm 4.8: s-Size-Constrained Min-k-Cut

where $\sigma_{k-1} := \sum_{i=1}^{k-1} s_i$.

Proof. We consider Algorithm 4.8. We first analyze its run-time. Each recursive call reduces the number of vertices in the hypergraph. Thus, the algorithm makes at most n recursive calls. Apart from the recursion steps, the algorithm only selects random partitions and performs contractions, both of which can be implemented to run in polynomial time.

Now we analyze the success probability. Let $\mathcal{G}_n$ be the set of all vertex-weighted n-vertex hypergraphs which contain an s-size-constrained k-cut. For a vertex-weighted hypergraph (G, w), let $M(G, w)$ be the set of all s-size-constrained min-k-cuts in (G, w). Define

$$q_n := \min_{(G,w)\in\mathcal{G}_n} \min_{F \in M(G,w)} \Pr[\text{Algorithm returns } F \text{ on input } (G, w, k, s)], \text{ and} \tag{4.72}$$

$$Q_n := \begin{cases} \left(k^{\max\{2\sigma_{k-1},\sigma_k\}}\right)^{-1} & \text{if } n \leq \max\{2\sigma_{k-1}, \sigma_k\}, \\ \left(k^{\max\{2\sigma_{k-1},\sigma_k\}} n \binom{n}{2\sigma_{k-1}}\right)^{-1} & \text{if } n > \max\{2\sigma_{k-1}, \sigma_k\}. \end{cases} \tag{4.73}$$

We note that $Q_n = \Omega(k^{-\sigma_{k-1}-\sigma_k} n^{-2\sigma_{k-1}-1})$, so it suffices to show that $q_n \geq Q_n$ for all $n \geq k$ (for smaller n, there are no k-cuts).

We proceed by induction on n. Let F be an s-size-constrained min-k-cut in (G, w). Let Y be an s-size-constrained k-partition with $F = \delta(Y)$. We assume that $|Y_1| \le |Y_2| \le \cdots \le |Y_k|$. This assumption is without loss of generality because we can relabel the parts of an s-size-constrained k-partition so that they are in increasing order of size and the resulting partition will still be an s-size-constrained k-partition (since we have assumed that the size vector s is in increasing order).

For the base case, suppose $n \le \max\{2\sigma_{k-1}, \sigma_k\}$. In this case, the algorithm returns $\delta(X)$ where X is a k-partition of V chosen uniformly at random. Since $F = \delta(Y)$, the probability that $F = \delta(X)$ for the X randomly chosen by the algorithm is at least $\Pr[X = Y]$. The number of k-partitions of V is at most k^n, the number of ways to assign each of the n vertices to one of k labeled sets. Thus, $\Pr[X = Y] \ge k^{-n} \ge k^{-max(2\sigma_{k-1}, \sigma_k)} = Q_n$.

Now we will prove the induction step. Assume that $n > \max\{2\sigma_{k-1}, \sigma_k\}$. By the induction hypothesis, we have $q_{n'} \ge Q_{n'}$ for all $n' \in \{k, \ldots, n-1\}$. We will show that $q_n \ge Q_n$.

Suppose $|Y_k| \ge n - 2\sigma_{k-1}$. Let T be an arbitrary subset of Y_k of size $n - 2\sigma_{k-1}$. Consider the set S chosen by the algorithm. The probability that S is equal to $V - T$ is $\binom{n}{2\sigma_{k-1}}^{-1}$. Next, consider the sets X_i created by the algorithm. The probability that $X_i = Y_i$ for every $i \in [k]$ conditioned on $S = V - T$, is $k^{-2\sigma_{k-1}}$. Thus, the probability that the k-partition X obtained in the algorithm is identical to Y is at least $k^{-2\sigma_{k-1}}\binom{n}{2\sigma_{k-1}}^{-1}$. Since the last step of the algorithm returns $R = \delta(X)$ with probability $1/n$, it follows that the algorithm returns F with probability at least $\left(nk^{\max\{2\sigma_{k-1}, \sigma_k\}}\binom{n}{2\sigma_{k-1}}\right)^{-1} = Q_n$.

Henceforth, we assume that $|Y_k| < n - 2\sigma_{k-1}$. We will call a hyperedge *large* if it contains at least $n - 2\sigma_{k-1}$ vertices. Since Y_k is the largest part of the k-partition Y, every large hyperedge must be contained in the k-cut F. In particular, if $\alpha_e = 0$ for a hyperedge e, then $\binom{n-|e|}{\sigma_{k-1}} = 0$, which implies that $n - |e| < \sigma_{k-1}$, and hence e is large and consequently, e cannot be in F.

Next, suppose that $\alpha_e = 0$ for every hyperedge e. Then every hyperedge is a large hyperedge and therefore, $F = E$. In this case, the algorithm will return R. We note that $R = E$ if X is not a k-partition. We lower bound the probability that X is not a k-partition now. If all vertices in S are assigned to X_k, then X is not a k-partition. The probability that all vertices in S are assigned to X_k is $k^{-2\sigma_{k-1}}$. Thus, the probability that the algorithm returns $F = R = E$ is at least $k^{-2\sigma_{k-1}} \ge Q_n$.

Henceforth, we assume that $\alpha_e > 0$ for some hyperedge $e \in E$. This means that the algorithm will contract some hyperedge and then recurse on the resulting hypergraph. Let e' be a random variable for the hyperedge chosen to be contracted. Let w' be the weight function defined on the vertices of G/e' as follows: $w'(v) := w(v)$ for each $v \in V \setminus e'$ and

$w'(v) := \sum_{u\in e'} w(u)$ where v is the new vertex resulting from the contraction. If $e' \notin F$, then F will be an s-size-constrained min-k-cut in $(G/e', w')$. Therefore, we have that

$$\Pr[R' = F \text{ on input } (G,k,s)] \tag{4.74}$$
$$= \sum_{e\in E\setminus F} \Pr[e' = e] \cdot \Pr[\text{Algorithm returns } F \text{ on input } (G/e,k,s)] \tag{4.75}$$
$$\geq \sum_{e\in E\setminus F} \frac{\alpha_e}{\sum_{f\in E}\alpha_f} \cdot q_{n-|e|+1}. \tag{4.76}$$

Let e be a hyperedge that is not in the k-cut F. Then, e cannot be a large hyperedge and hence, $|e| < n - 2\sigma_{k-1}$. Consequently, $n - |e| + 1 > 2\sigma_{k-1} + 1 \geq k$. Therefore, by applying the induction hypothesis, we have $q_{n-|e|+1} \geq Q_{n-|e|+1}$. Hence,

$$\Pr[R' = F \text{ on input } (G,k,s)] \geq \frac{1}{\sum_{f\in E}\alpha_f} \sum_{e\in E\setminus F} \alpha_e \cdot Q_{n-|e|+1}. \tag{4.77}$$

We need the following two claims. We defer their proofs to complete the proof of the theorem.

Claim 4.6. For every hyperedge $e \in E \setminus F$, we have $\alpha_e Q_{n-|e|+1} \geq \frac{nQ_n}{n-1}$.

Claim 4.7. $\frac{|E\setminus F|}{\sum_{f\in E}\alpha_f} \geq 1$.

By Claim 4.6, we have that

$$\Pr[R' = F \text{ on input } (G,k,s)] = \frac{1}{\sum_{f\in E}\alpha_f} \sum_{e\in E\setminus F} \alpha_e \cdot Q_{n-|e|+1} \tag{4.78}$$
$$\geq \frac{1}{\sum_{f\in E}\alpha_f} \sum_{e\in E\setminus F} \frac{nQ_n}{n-1} \tag{4.79}$$
$$= \frac{|E\setminus F|}{\sum_{f\in E}\alpha_f} \cdot \frac{nQ_n}{n-1}. \tag{4.80}$$

Thus, Claim 4.7 implies that

$$\Pr[R' = F \text{ on input } (G,k,s)] \geq \frac{nQ_n}{n-1}. \tag{4.81}$$

Finally, we note that since we have assumed $n > \max\{2\sigma_{k-1}, \sigma_k\}$ and $\alpha_e > 0$ for some e, the probability that the algorithm returns R' is $(n-1)/n$. Thus, we conclude that $\Pr[\text{Algorithm returns } F] \geq Q_n$. QED.

Proof of Claim 4.6. Let $e \in E\setminus F$. We recall that F contains all large hyperedges and hence,

$$|e| < n - 2\sigma_{k-1}. \tag{4.82}$$

First, suppose that $n-|e|+1 \le \max\{2\sigma_{k-1}, \sigma_k\}$. Then, we have $Q_{n-|e|+1} = k^{-\max\{2\sigma_{k-1},\sigma_k\}}$. We consider two subcases.

Case 1: Suppose $n \ge 3\sigma_{k-1}$. Then

$$\frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}} Q_{n-|e|+1} \ge \frac{Q_{n-|e|+1}}{\binom{n}{\sigma_{k-1}}} \ge \left(k^{\max\{2\sigma_{k-1},\sigma_k\}}\binom{n}{2\sigma_{k-1}}\right)^{-1} = nQ_n \ge \frac{nQ_n}{n-|e|+1}. \tag{4.83}$$

The first inequality follows from inequality (4.82), and the last inequality follows from the fact that $\binom{n}{\sigma_{k-1}} \le \binom{n}{2\sigma_{k-1}}$ for $n \ge 3\sigma_{k-1}$.

Case 2: Suppose $n < 3\sigma_{k-1}$. We recall that $n > 2\sigma_{k-1}$. By inequality (4.82), $n-|e| > 2\sigma_{k-1}$. Letting $n = 2\sigma_{k-1} + x$ for some $x \in \{1, \dots \sigma_{k-1}\}$, we have

$$\frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}} \ge \frac{\binom{2\sigma_{k-1}}{\sigma_{k-1}}}{\binom{2\sigma_{k-1}+x}{\sigma_{k-1}}} = \frac{(2\sigma_{k-1})!(\sigma_{k-1}+x)!}{(2\sigma_{k-1}+x)!\sigma_{k-1}!} = \prod_{i=1}^{x} \frac{\sigma_{k-1}+i}{2\sigma_{k-1}+i} \ge \left(\frac{1}{2}\right)^{x}. \tag{4.84}$$

We also know that

$$\binom{n}{2\sigma_{k-1}}^{-1} = \binom{2\sigma_{k-1}+x}{2\sigma_{k-1}}^{-1} = \frac{(2\sigma_{k-1})!x!}{(2\sigma_{k-1}+x)!} = \prod_{i=1}^{x} \frac{i}{2\sigma_{k-1}+i} \le \left(\frac{1}{2}\right)^{x}. \tag{4.85}$$

Therefore,

$$\frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}} Q_{n-|e|+1} \ge \left(\frac{1}{2}\right)^{x} Q_{n-|e|+1} \tag{4.86}$$

$$\ge \frac{Q_{n-|e|+1}}{\binom{n}{2\sigma_{k-1}}} \tag{4.87}$$

$$= \left(k^{\max\{2\sigma_{k-1},\sigma_k\}}\binom{n}{2\sigma_{k-1}}\right)^{-1} \tag{4.88}$$

$$= nQ_n \tag{4.89}$$

$$\ge \frac{nQ_n}{n-|e|+1}. \tag{4.90}$$

Next, suppose that $n - |e| + 1 > \max\{2\sigma_{k-1}, \sigma_k\}$. We have that

$$\begin{aligned}
\alpha_e Q_{n-|e|+1} &= \frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}} Q_{n-|e|+1} && (4.91)\\
&= \frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}} \cdot \frac{1}{\binom{n-|e|+1}{2\sigma_{k-1}}} \cdot \left((n-|e|+1)k^{\max\{2\sigma_{k-1},\sigma_k\}}\right)^{-1} && (4.92)\\
&= \frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}} \cdot \frac{1}{\binom{n-|e|+1}{2\sigma_{k-1}}} \cdot \frac{n}{n-|e|+1}\binom{n}{2\sigma_{k-1}} \cdot Q_n && (4.93)\\
&\geq \frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}} \cdot \frac{1}{\binom{n-|e|+1}{2\sigma_{k-1}}} \cdot \binom{n}{2\sigma_{k-1}} \cdot \frac{nQ_n}{n-1}. && (4.94)
\end{aligned}$$

The following proposition completes the proof. QED.

Proposition 4.1. For positive integers n, e, σ with $e \geq 2$ and $n - e + 1 > 2\sigma$, we have

$$\frac{\binom{n-e}{\sigma}}{\binom{n}{\sigma}} \cdot \frac{1}{\binom{n-e+1}{2\sigma}} \geq \binom{n}{2\sigma}^{-1}.$$

Proof. We note that

$$\begin{aligned}
\frac{\binom{n-e}{\sigma}}{\binom{n}{\sigma}} \cdot \frac{1}{\binom{n-e+1}{2\sigma}} &= \frac{(n-e)!(n-\sigma)!}{n!(n-e-\sigma)!} \cdot \frac{(2\sigma)!(n-e-2\sigma+1)!}{(n-e+1)!} && (4.95)\\
&= \left(\prod_{i=0}^{\sigma-1} \frac{n-e-i}{n-i}\right) \cdot \frac{(2\sigma)!}{\prod_{i=0}^{2\sigma-1}(n-e+1-i)}. && (4.96)
\end{aligned}$$

To lower bound this expression, we case on the value of e.

Case 1: Suppose $e > \sigma$. Then we can lower bound expression (4.96) by

$$\frac{1}{\prod_{i=0}^{\sigma-1}(n-i)} \cdot \frac{(2\sigma)!}{(n-e+1)\prod_{i=0}^{\sigma-2}(n-e-\sigma-i)} \geq \frac{(2\sigma)!}{\prod_{i=0}^{2\sigma-1}(n-i)} = \binom{n}{2\sigma}^{-1}. \tag{4.97}$$

Case 2: Suppose $e \leq \sigma$. We note that

$$\frac{(2\sigma)!}{\prod\limits_{i=0}^{2\sigma-1}(n-e+1-i)} = \frac{(2\sigma)!}{\prod\limits_{i=0}^{2\sigma-1}(n-i)} \cdot \prod_{i=0}^{e-2} \frac{n-i}{n-2\sigma-i} = \binom{n}{2\sigma}^{-1} \cdot \prod_{i=0}^{e-2} \frac{n-i}{n-2\sigma-i}. \tag{4.98}$$

Thus, expression (4.96) is equal to

$$\binom{n}{2\sigma}^{-1} \cdot \left(\prod_{i=0}^{\sigma-1} \frac{n-e-i}{n-i}\right) \left(\prod_{i=0}^{e-2} \frac{n-i}{n-2\sigma-i}\right). \tag{4.99}$$

We will show that $\left(\prod_{i=0}^{\sigma-1} \frac{n-e-i}{n-i}\right) \left(\prod_{i=0}^{e-2} \frac{n-i}{n-2\sigma-i}\right) \geq 1$. We note that

$$\begin{aligned}\left(\prod_{i=0}^{\sigma-1} \frac{n-e-i}{n-i}\right) \left(\prod_{i=0}^{e-2} \frac{n-i}{n-2\sigma-i}\right) &= \frac{\prod_{i=0}^{\sigma-1}(n-e-i)}{\prod_{i=e-1}^{\sigma-1}(n-i)} \cdot \frac{1}{\prod_{i=0}^{e-2}(n-2\sigma-i)} \\ &= \frac{\prod_{i=\sigma}^{e+\sigma-1}(n-i)}{(n-e+1)\prod_{i=0}^{e-2}(n-2\sigma-i)}. \end{aligned} \tag{4.100}$$

We claim that expression (4.100) is minimized when $e = 2$. To see this, we note that

$$\frac{\displaystyle\prod_{i=\sigma}^{(e+1)+\sigma-1}(n-i)}{(n-(e+1)+1)\displaystyle\prod_{i=0}^{(e+1)-2}(n-2\sigma-i)} = \frac{\displaystyle\prod_{i=\sigma}^{e+\sigma-1}(n-i)}{(n-e+1)\displaystyle\prod_{i=0}^{e-2}(n-2\sigma-i)} \cdot \frac{(n-e-\sigma)(n-e+1)}{(n-2\sigma-e+1)(n-e)}. \tag{4.101}$$

Since $e \leq \sigma$, we know that $n-e-\sigma \geq n-2\sigma+1$. From this, along with the fact that $n-e+1 > n-e$, we conclude that $\frac{(n-e-\sigma)(n-e+1)}{(n-2\sigma-e+1)(n-e)} > 1$. This means that expression (4.100) increases when we increment e. Thus

$$\frac{\prod_{i=\sigma}^{e+\sigma-1}(n-i)}{(n-e+1)\prod_{i=0}^{e-2}(n-2\sigma-i)} \geq \frac{(n-\sigma)(n-\sigma-1)}{(n-1)(n-2\sigma)} = \frac{n^2-(2\sigma+1)n+(\sigma+1)\sigma}{n^2-(2\sigma+1)n+2\sigma} \geq 1. \tag{4.102}$$

The last inequality follows from the fact that $\sigma \geq 1$.

Thus, we have shown that $\left(\prod_{i=0}^{\sigma-1} \frac{n-e-i}{n-i}\right) \left(\prod_{i=0}^{e-2} \frac{n-i}{n-2\sigma-i}\right) \geq 1$, and therefore, combining the above inequalities, we have that

$$\frac{\binom{n-e}{\sigma}}{\binom{n}{\sigma}} \cdot \frac{1}{\binom{n-e+1}{2\sigma}} \geq \binom{n}{2\sigma}^{-1}. \tag{4.103}$$

QED.

Proof of Claim 4.7. Let $Z = (Z_1, \ldots, Z_k)$ be a random k-partition obtained by picking disjoint sets $Z_1, \ldots, Z_{k-1}$ with $|Z_i| = s_i$ and setting $Z_k = V \setminus \bigcup_{i=1}^{k-1} Z_i$. Since $n > \sigma_k$ and every vertex has weight at least 1, the k-partition Z is an s-size-constrained k-partition.

Therefore, $|\delta(Z)|$ is an upper bound on $|F|$. In particular,

$$|F| \leq \mathbb{E}(|\delta(Z)|) = \sum_{e\in E} \Pr(e \in \delta(Z)). \tag{4.104}$$

Negating the inequality and adding $|E|$ to both sides gives

$$\begin{aligned}
|E \setminus F| &\geq \sum_{e\in E}(1 - \Pr(e \in \delta(Z))) && (4.105)\\
&= \sum_{e\in E} \Pr(e \notin \delta(Z)) && (4.106)\\
&= \sum_{e\in E}\sum_{i=1}^{k} \Pr(e \subseteq Z_i) && (4.107)\\
&\geq \sum_{e\in E} \frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}} && (4.108)\\
&= \sum_{e\in E} \alpha_e. && (4.109)
\end{aligned}$$

Thus, $|E \setminus F| / \sum_{f\in E} \alpha_f \geq 1$. QED.

Remark 4.3 Since our algorithm does not even take the vertex-weights as input, it could trivially be extended to handle a version of the problem where we have multiple weight functions on the vertices (as in the previous sections) each with their own minimum sizes. If we have t vertex-weight functions, $w_1, \ldots, w_t : V \to \mathbb{Z}_+$ and each function w_j has an associated list of lower bounds $s_{j,1}, \ldots, s_{j,k}$, then we can find a min-k-cut satisfying all of these lower-bound constraints with at least inverse polynomial probability by simply running our algorithm with $s_i = \max_{j\in[t]} s_{j,i}$ for every $i \in [k]$.

4.5 COMPARISON OF PARAMETRIC, PARETO-OPTIMAL, AND MULTIOBJECTIVE CUTS

We prove the containment relationship (4.1) here.

Proposition 4.2. The following containment relationship holds, possibly with the containment being strict:

$$\text{Parametric min-cuts} \subseteq \text{Pareto-optimal cuts} \subseteq \text{Multiobjective min-cuts}.$$

Proof. We first show that parametric min-cuts are pareto-optimal cuts: If a cut F' dominates a cut F, then $w(F') < w(F)$ for all positive multipliers, and therefore F cannot be a parametric min-cut. On the other hand, not every pareto-optimal cut is a parametric min-cut (see Figure 4.2 for an example).

Next, we show that pareto-optimal cuts are multiobjective min-cuts: If a cut F is pareto-optimal, then it is a b-multiobjective min-cut for the budget-vector b obtained by setting $b_i := c_i(F)$ for every $i \in [k-1]$. On the other hand, not every multiobjective min-cut is a pareto-optimal cut (see Figure 4.3 for an example).

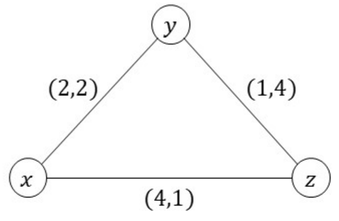

Figure 4.2: The cut $\delta(z)$ is a pareto-optimal cut but not a parametric min-cut.

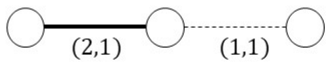

Figure 4.3: For $b = 2$, the bold edge is a b-multiobjective min-cut but it is not a pareto-optimal cut.

QED.

4.6 CONCLUSION AND OPEN PROBLEMS

In this chapter, we illustrated the versatility of the random contraction technique by addressing multicriteria versions of min-cut and size-constrained min-k-cut problems. There are several interesting open questions in this area. We conclude by stating a few:

1. For the number of pareto-optimal cuts and multiobjective min-cuts, there is still a gap between our lower bound (which is $\Omega(n^t)$) and our upper bound (which is $O(n^{3t-1})$). Can we improve either of these bounds? We believe that improving our bounds for the number of b-multiobjective min-cuts for a fixed budget-vector $b \in \mathbb{R}_{\geq 0}^{t-1}$ would be a first-step towards this goal.

2. We gave a polynomial-time algorithm to solve the b-multiobjective min-cut problem in constant-rank hypergraphs. How about arbitrary-rank hypergraphs? Is the b-multiobjective min-cut problem in arbitrary rank hypergraphs (even for $t = 2$ criteria) solvable in polynomial-time or is it NP-hard?

CHAPTER 5: APPROXIMATE REPRESENTATION OF SYMMETRIC SUBMODULAR FUNCTIONS USING HYPERGRAPH CUT FUNCTIONS

Submodular functions are fundamental to combinatorial optimization. Many interesting problems can be formulated as special cases of problems involving submodular functions. In this chapter, we consider the problem of approximating symmetric submodular functions everywhere using hypergraph cut functions. Devanur, Dughmi, Schwartz, Sharma, and Singh [88] showed that symmetric submodular functions over n-element ground sets cannot be approximated within a factor of $n/8$ using a graph cut function and raised the question of approximating them using hypergraph cut functions. Our main result is that there exist symmetric submodular functions over n-element ground sets that cannot be approximated within a factor of $o(n^{1/3}/\log^2 n)$ using a hypergraph cut function. On the positive side, we show that symmetrized concave linear functions and symmetrized rank functions of uniform matroids and partition matroids can be constant-approximated using hypergraph cut functions. The results in this chapter are based on joint work with Chandrasekaran, Chekuri, and Xu and appeared in FSTTCS '22 [92]. The research in this chapter was supported in part by NSF grants CCF-1814613 and CCF-1907937.

5.1 INTRODUCTION

A set function $f : 2^V \to \mathbb{R}_{\geq 0}$ defined over a ground set V is *submodular* if $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$ for all subsets $A, B \subseteq V$ and is *symmetric* if $f(A) = f(V - A)$ for all subsets $A \subseteq V$. Submodular functions have the diminishing marginal returns property which arises frequently in economic and game theoretic contexts. Well-known examples of submodular functions include matroid rank functions and graph/hypergraph cut functions. Owing to these connections, submodular functions play a fundamental role in combinatorial optimization.

Throughout this work, we will be interested in non-negative set functions $f : 2^V \to \mathbb{R}_{\geq 0}$ with $f(\emptyset) = 0$. We use n to denote the size of the ground set V. Describing a submodular function by listing out $f(S)$ for every $S \subseteq V$ requires a string of $\Omega(2^n)$ bits, even if f only outputs values in $\{0, 1\}$. Given the prevalence of submodular functions in combinatorial optimization, a natural question is whether there exists a concise representation—i.e. using $2^{o(n)}$ bits—for arbitrary symmetric submodular functions. Unfortunately, it is impossible to construct such a representation using $2^{o(n)}$ function evaluation queries [140], and in fact concise representations for arbitrary symmetric submodular functions do not exist. In Section 5.1.3 we give a short proof that exact representation of symmetric submodular functions (whose outputs are non-negative integers bounded by n) using $2^{o(n)}$ bits is impossible.

For a parameter $\alpha \geq 1$, a set function $g : 2^V \to \mathbb{R}_{\geq 0}$ is said to α-approximate a set function $f : 2^V \to \mathbb{R}_{\geq 0}$ if

$$g(A) \leq f(A) \leq \alpha g(A) \ \forall \ A \subseteq V. \tag{5.1}$$

Given that exactly representing a submodular function requires a string of $\Omega(2^n)$ bits, the next natural question which has been studied is whether an arbitrary submodular set function can be well-approximated by a concisely representable function. We distinguish between structural and algorithmic variants of this question: the structural question asks whether submodular functions can be well-approximated via concisely representable functions while the algorithmic question asks whether such a concise representation can be constructed using polynomial number of function evaluation queries (note that the algorithmic question is concerned with the number of function evaluation queries as opposed to run-time). Concise representations with small-approximation factor are useful in learning, testing, streaming, and sketching algorithms. Consequently, concise representations with small-approximation factor for submodular functions (and their generalizations and subfamilies of submodular functions) have been studied from all these perspectives with most results focusing on monotone submodular functions [84, 85, 86, 87, 88, 89, 90, 91].

In this chapter, we focus on approximating *symmetric* submodular functions. Balcan, Harvey, and Iwata [85] showed that for every symmetric submodular function $f : 2^V \to \mathbb{R}_{\geq 0}$, there exists a function $g : 2^V \to \mathbb{R}_{\geq 0}$ defined by $g(S) := \sqrt{\chi(S)^T M \chi(S)}$, where $\chi(S) \in \{0,1\}^V$ is the indicator vector of $S \subseteq V$ and M is a symmetric positive definite matrix such that g $\sqrt{n}$-approximates f. We note that such a function g has a concise representation—namely, the matrix M. Is it possible to improve on the approximation factor for symmetric submodular functions using other concisely representable functions?

The concisely representable family of functions that we study in this chapter is the family of hypergraph cut functions. A hypergraph $H = (V, E)$ consists of a vertex set V and hyperedges E where each hyperedge $e \in E$ is a subset of vertices. If every hyperedge has size 2, then the hypergraph is simply a graph. For a subset A of vertices, we use $\delta(A)$ to denote the set of hyperedges e such that e has non-empty intersection with both A and $V \setminus A$. The cut function $d : 2^V \to \mathbb{R}_+$ of a hypergraph $H = (V, E)$ with hyperedge weights $w : E \to \mathbb{R}_+$ is given by

$$d(A) := \sum_{e \in E:\ e \in \delta(A)} w_e \ \forall \ A \subseteq V. \tag{5.2}$$

A function $g : 2^V \to \mathbb{R}_+$ is a hypergraph cut function if there exists a weighted hypergraph with vertex set V whose cut function is g. We will say that a function $f : 2^V \to \mathbb{R}_+$ is

α-hypergraph-approximable (α-graph approximable) if there exists a hypergraph (graph) cut function g such that g α-approximates f. We note that although a hypergraph could have exponential number of hyperedges, every n-vertex hypergraph admits a $(1+\epsilon)$-approximate cut-sparsifier with $O(\frac{n \log n}{\epsilon^2})$ hyperedges (see Theorem 5.4 for a formal definition of cut-sparsifier), and hence, hypergraph cut functions have a concise representation (with a constant loss in approximation factor).

The structural approximation question of whether every symmetric submodular function is constant-hypergraph-approximable was raised by Devanur, Dughmi, Shwartz, Sharma, and Singh [88]. They showed that every symmetric submodular function on a ground set of size n is $O(n)$-graph-approximable and that this factor is tight for graph-approximability: in fact, the cut function of the n-vertex hypergraph containing a single hyperedge that contains all vertices cannot be $(n/4-\epsilon)$-approximated by a graph cut function for all constant $\epsilon > 0$. This example naturally raises the following intriguing conjecture:

Conjecture 5.1. Every symmetric submodular function is $O(1)$-hypergraph-approximable.

The conjecture is further fueled by the fact that there are no natural examples of symmetric submodular functions besides hypergraph cut functions (although arbitrary submodular functions can be symmetrized while preserving submodularity).

We emphasize that the algorithmic variant of Conjecture 5.1 is false. In particular, there does not exist an algorithm that makes a polynomial number of function evaluation queries to a symmetric submodular function f and constructs a hypergraph cut function g such that g $O(\sqrt{n/\ln n})$-approximates f. We outline a proof of this observation now. Suppose that there exists an algorithm that uses polynomial number of function evaluation queries to a given symmetric submodular function f to construct a weighted hypergraph whose cut function α-approximates f; then we can obtain an α-approximation to the *symmetric submodular sparsest cut problem* by constructing such a hypergraph and solving the sparsest cut on that hypergraph exactly (using exponential run-time). However, Svitkina and Fleischer [86] have shown that the best possible approximation for the symmetric submodular sparsest cut problem using polynomial number of function evaluation queries is $\Omega(\sqrt{n/\ln n})$ (even if exponential run-time is allowed). Hence, the algorithmic version of hypergraph-aproximability has a strong lower bound of $\Omega(\sqrt{n/\ln n})$. This leaves the structural question open while perhaps, hinting that it may also have a strong lower bound.

5.1.1 Our Results

The symmetrization of a set function $f : 2^V \to \mathbb{R}$ is the function $f_{\text{sym}} : 2^V \to \mathbb{R}$ obtained as

$$f_{\text{sym}}(A) := f(A) + f(V \setminus A) - f(V) - f(\emptyset). \tag{5.3}$$

We note that if $f : 2^V \to \mathbb{R}$ is submodular, then its symmetrization $f_{\text{sym}} : 2^V \to \mathbb{R}$ is symmetric submodular. A matroid rank function is a non-negative integer valued submodular set function $r : 2^V \to \mathbb{Z}$ satisfying $r(A) \le r(A \cup \{e\}) \le r(A) + 1$ for every subset $A \subseteq V$ and element $e \in V$. As a step towards understanding Conjecture 5.1, we observe that it suffices to focus on symmetrized matroid rank functions (see Section 5.1.4 for a proof).

Proposition 5.1. If the symmetrization of every matroid rank function is α-hypergraph-approximable, then every rational-valued symmetric submodular function is α-hypergraph-approximable.

Next, we refute Conjecture 5.1 by showing the following result.

Theorem 5.1. For every sufficiently large positive integer n, there exists a matroid rank function $r : 2^{[n]} \to \mathbb{Z}_{\ge 0}$ such that r_{sym} is not α-hypergraph-approximable for

$$\alpha = o\left(\frac{n^{\frac{1}{3}}}{\log^2 n}\right).$$

Our proof of Theorem 5.1 is an existential argument and it does not construct an explicit matroid rank function that achieves the lower bound.

Next, we prove positive approximation results for certain subfamilies of symmetric submodular functions. The subfamilies that we consider are inspired by Proposition 5.1 and by previous work on approximating symmetric submodular functions and matroid rank functions.

We call a set function $f : 2^V \to \mathbb{R}_{\ge 0}$ a *concave linear function* if there exist weights $w : V \to \mathbb{R}_{\ge 0}$ and an increasing concave function $h : \mathbb{R}_{\ge 0} \to \mathbb{R}_{\ge 0}$ such that $f(S) = h(\sum_{v \in S} w_v)$ for every $S \subseteq V$. We note that concave linear functions are submodular. Goemans, Harvey, Iwata, and Mirrokni [84] showed that every matroid rank function over a n-element ground set can be $\sqrt{n}$-approximated by the square-root of a linear function, i.e., by a concave linear function. Balcan, Harvey and Iwata [85] showed that every symmetric submodular function $f : 2^V \to \mathbb{R}_{\ge 0}$ is $\sqrt{n}$-approximated by a function $g : 2^V \to \mathbb{R}_{\ge 0}$ of the form $g(S) := \sqrt{\chi(S)^T M \chi(S)}$ for all $S \subseteq V$, where $\chi(S) \in \{0, 1\}^V$ is the indicator vector of S and M is a symmetric positive definite matrix. In particular, if M is a diagonal matrix, then the function g is the square root of a linear function, i.e., a concave linear function. Given the significant role of concave linear functions, we consider the hypergraph-approximability of such functions.

Theorem 5.2. Symmetrized concave linear functions are 128-hypergraph-approximable.

As a special case of Theorem 5.2, we obtain that the symmetrized rank functions of uniform matroids are constant-hypergraph-approximable. Thus, symmetrized rank functions of uniform matroids act as a starting point for identifying subfamilies of symmetrized matroid rank functions that are constant-hypergraph-approximable. We consider a generalization of the uniform matroid, namely the partition matroid, and show that it is also constant-hypergraph-approximable. We refer the reader to Section 5.1.2 for formal definitions of uniform and partition matroids.

Theorem 5.3. Symmetrized rank functions of uniform matroids and partition matroids are 64-hypergraph-approximable.

Theorem 5.3 gives a concrete class of functions for which there is a large gap between the approximation capabilities of graph cut functions and hypergraph cut functions. Consider the uniform matroid where the independent sets are those of size at most 1. The symmetrized rank function of this matroid is the same as the cut function of a hypergraph with a single hyperedge spanning all vertices. As mentioned above, this function cannot be $(n/4 - \epsilon)$-approximated by a graph cut function for all constant $\epsilon > 0$ [88]. Thus, symmetrized rank functions of uniform and partition matroids cannot be better than $n/4$ approximated by graph cut functions, but can be constant factor approximated by hypergraph cut functions.

While our lower bound result in Theorem 5.1 rules out α-hypergraph-approximability for symmetric submodular functions for $\alpha = o(n^{1/3}/\log^2 n)$, our positive results suggest broad families of symmetric submodular functions which are constant-hypergraph-approximable. It would be interesting to characterize the family of symmetric submodular functions that are constant-hypergraph-approximable. We also do not know if our lower bound result in Theorem 5.1 is tight. We only know that every symmetric submodular function is $(n-1)$-graph-approximable. It would be interesting to show that every symmetric submodular function is $\tilde{O}(n^{1/3})$-hypergraph-approximable—we believe that Proposition 5.1 and Theorem 5.2 should help towards achieving this approximation factor.

5.1.2 Preliminaries

A set function $f : 2^V \to \mathbb{R}_{\geq 0}$ is submodular if $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$ for all subsets $A, B \subseteq V$, symmetric if $f(A) = f(V - A)$ for all subsets $A \subseteq V$, and monotone if

$f(B) \geq f(A)$ for all subsets $A \subseteq B \subseteq V$.

A matroid $\mathcal{M} = (V, \mathcal{I})$ is specified by a ground set V and a collection $\mathcal{I} \subseteq 2^V$, known as independent sets, satisfying the three independent set axioms: (1) $\emptyset \in \mathcal{I}$, (2) if $B \in \mathcal{I}$, then $A \in \mathcal{I}$ for every $A \subseteq B$, and (3) if $A, B \in \mathcal{I}$ with $|B| > |A|$, then there exists an element $v \in B \setminus A$ such that $A \cup \{v\} \in \mathcal{I}$. The rank function $r : 2^V \to \mathbb{Z}_{\geq 0}$ of a matroid $\mathcal{M} = (V, \mathcal{I})$ is defined as

$$r(A) := \max\{|S| : S \subseteq A,\ S \in \mathcal{I}\}\ \forall\ A \subseteq V. \tag{5.4}$$

The definition of matroid rank functions that we presented in Section 5.1.1 is equivalent to this definition [141]. It is well-known that the rank function of a matroid is monotone submodular.

We consider two matroids over the ground set V. A uniform matroid is a matroid in which the independent sets are exactly the sets containing at most k elements of the ground set V, for some fixed integer k—we call it as the uniform matroid over ground set V with budget k. Partition matroids generalize uniform matroids: the independent sets of the partition matroid associated with a partition $P_1, \ldots, P_t$ of the ground set V with budgets $b_1, \ldots, b_t \in \mathbb{Z}_{\geq 0}$ are those subsets $A \subseteq V$ for which $|A \cap P_i| \leq b_i$ for every $i \in [t]$.

The proof of our lower bound will use the following theorem showing the existence of cut-sparsifiers.

Theorem 5.4. [142] For every positive constant ϵ and for every weighted n-vertex hypergraph H, there exists another weighted hypergraph H' (called a *cut-sparsifier*) on the same vertex set with $\tilde{O}(n/\epsilon^2)$ hyperedges such that the cut function of H' $(1+\epsilon)$-approximates the cut function of H.

5.1.3 Impossibility of Exact Concise Representation of Symmetric Submodular Functions

In this section we motivate our focus on approximate representation by showing that symmetric submodular functions which take on non-negative integer values bounded by n do not admit an exact representation of size $2^{o(n)}$, where n is the size of the ground set. In particular, we prove Theorem 5.5. Theorem 5.5 implies that it is impossible to exactly represent arbitrary symmetric submodular functions over ground set $[n]$ which take on non-negative integer values at most n using $2^{o(n)}$ bits, because the number of $2^{o(n)}$-bit strings is $2^{2^{o(n)}}$, while the number of distinct symmetric submodular functions over ground set $[n]$ which take on non-negative integer values at most n is $2^{2^{\Omega(n)}}$.

Theorem 5.5. For every even integer $n \geq 2$, there exists a family $\mathcal{F}$ of $2^{2^{\Omega(n)}}$ symmetric submodular functions from $2^{[n]}$ to $\{0,\dots,n\}$ such that for every $f,g \in \mathcal{F}$, there exists a set $S \subseteq [n]$ such that $f(S) \neq g(S)$.

Proof. Let $V := [n]$. Let $U := \{S \subseteq [n] \colon |S| = n/2\}$. Let $\mathcal{G}$ be the set of all symmetric functions $g : U \to \{0,1\}$. We note that $|\mathcal{G}|$ is $2^{|U|/2}$. For every function $g \in \mathcal{G}$, we define a function $f_g : 2^V \to \{0,\dots,n\}$ by

$$f_g(S) = \begin{cases} \min\{|S|, |V \setminus S|\} & \text{if } |S| \neq n/2 \\ n/2 - g(S) & \text{otherwise.} \end{cases} \tag{5.5}$$

Claim 5.1. For every $g \in \mathcal{G}$, f_g is symmetric and submodular.

Proof. Let $g \in \mathcal{G}$ be arbitrary. We first show that f_g is symmetric. If $|S| \neq n/2$ then $f_G(S) = f_G(V \setminus S) = \min\{|S|, |V \setminus S|\}$. If $|S| = n/2$ then, because g is symmetric, we have that $f_g(S) = n/2 - g(S) = n/2 - g(V \setminus S) = f_g(V \setminus S)$. Therefore, for every $S \subseteq V$, $f_g(S) = f_g(V \setminus S)$, so f_g is symmetric.

Now we show that f_g is submodular. We note that for every $x \in V$ and every $S \subseteq V \setminus \{x\}$, we have that

$$f_g(S \cup \{x\}) - f_g(S) = \begin{cases} 1 & \text{if } |S| < n/2 - 1 \\ 0 \text{ or } 1 & \text{if } |S| = n/2 - 1 \\ 0 \text{ or } -1 & \text{if } |S| = n/2 \\ -1 & \text{if } |S| > n/2. \end{cases} \tag{5.6}$$

Thus, for any two sets $S, T \subseteq V \setminus \{x\}$ with $|S| < |T|$ we have that $f_g(S \cup \{x\}) - f_g(S) \geq f_g(T \cup \{x\}) - f_g(T)$. In particular, this inequality holds whenever $S \subsetneq T \subseteq V \setminus \{x\}$, so we conclude that f_g is submodular. QED.

Let $\mathcal{F} := \{f_g \colon g \in \mathcal{G}\}$. By Claim 5.1, we have that $\mathcal{F}$ is a family of symmetric submodular functions from $2^{[n]}$ to $[n]$. Furthermore, for every two distinct functions $g_1, g_2 \in \mathcal{G}$, there exists some $S \subseteq V$ such that $g_1(S) \neq g_2(S)$. For this same S we have that $f_{g_1}(S) \neq f_{g_2}(S)$, so all of the functions in $\mathcal{F}$ are distinct. Finally, we observe that

$$|\mathcal{F}| = |\mathcal{G}| = 2^{\frac{1}{2}\binom{n}{n/2}} = 2^{2^{\Omega(n)}}. \tag{5.7}$$

QED.

5.1.4 Proof of Proposition 5.1

In this section, we prove Proposition 5.1. We need the notion of contraction of set functions and hypergraphs. For a set function $f : 2^V \to \mathbb{R}_{\geq 0}$ and a subset A, the function $g : 2^{V-A+a} \to \mathbb{R}_{\geq 0}$ obtained by contracting f with respect to A is defined as

$$g(S) := \begin{cases} f(S) \text{ if } a \notin S, \\ f(S - a + A) \text{ if } a \in S. \end{cases} \tag{5.8}$$

If $f : 2^V \to \mathbb{R}_{\geq 0}$ is a symmetric submodular function and $A \subseteq V$, then the function obtained by contracting f with respect to A is also symmetric submodular. Let $H = (V, E)$ be a hypergraph with hyperedge weights $w : E \to \mathbb{R}_{\geq 0}$ and $A \subseteq V$. Then, the hypergraph obtained by contracting H with respect to A is defined as $H' = (V - A + a, E')$ where a is a new vertex not present in V and

$$E' := \{e - A + a : e \in E, e \cap A \neq \emptyset\} \cup \{e : e \in E, e \cap A = \emptyset\}. \tag{5.9}$$

We note that E' could have self-loops and that there is a surjection $\phi : E \to E'$ mapping each hyperedge to the hyperedge it is contracted into (which could be the same as the original hyperedge). We use this surjection to define the weight $w' : E' \to \mathbb{R}_{\geq 0}$ of hyperedges in E' as $w'(e') = \sum_{e \in E:\, \phi(e) = e'} w(e)$. We note that if f is the cut function of a weighted hypergraph $H = (V, E)$ with hyperedge weights $w : E \to \mathbb{R}_{\geq 0}$ and $A \subseteq V$, then the contraction of f with respect to A corresponds to the cut function of the weighted hypergraph (H', w') obtained by contracting H with respect to A. This leads to the following observation:

Observation 5.1. The contraction of a α-hypergraph-approximable function $f : 2^V \to \mathbb{R}_{\geq 0}$ with respect to a subset $A \subseteq V$ is also α-hypergraph-approximable.

Proposition 5.1. If the symmetrization of every matroid rank function is α-hypergraph-approximable, then every rational-valued symmetric submodular function is α-hypergraph-approximable.

Proof. It suffices to consider integer-valued symmetric submodular functions (multiply all function values by the product of the denominators of their rational expressions). Hence, we will focus on approximating integer-valued symmetric submodular functions.

Let $f : 2^V \to \mathbb{Z}_{\geq 0}$ be an integer-valued symmetric submodular function. It is known that there exists a vector $w \in \mathbb{R}^V$ such that the function $g : 2^V \to \mathbb{R}_{\geq 0}$ defined by

$$g(S) := f(S) + \sum_{u \in S} w_u \;\forall\; S \subseteq V \tag{5.10}$$

is integer-valued, monotone, and submodular [143, Section 3.3] (e.g., for our purposes, we can simply choose $w_u := \max\{f(S) \colon S \subseteq V\}$ for every $u \in V$). Since $f(V) = 0$, we have that $g(V) = \sum_{u \in V} w_u$. Consequently, $(1/2)g_{\text{sym}}(S) = (1/2)(g(S) + g(V - S) - g(V)) = (1/2)(f(S) + f(V - S)) = f(S)$ for every $S \subseteq V$ since f is symmetric. Thus, $f(S) = (1/2)g_{\text{sym}}(S)$ for every $S \subseteq V$.

Next, consider the integer-valued monotone submodular function $g : 2^V \to \mathbb{Z}_{\geq 0}$ obtained as above. Helgason [144] showed that there exists a matroid on a ground set U with rank function $r : 2^U \to \mathbb{Z}_{\geq 0}$ and a partition $(U_v : v \in V)$ of U such that $g(S) = r(\cup_{v \in S} U_v)$ for every $S \subseteq V$. Equivalently, the function g is obtained from the rank function r by repeatedly contracting with respect to U_v for each $v \in V$ (the order of processing $v \in V$ is irrelevant). Moreover, $f(S) = (1/2)g_{\text{sym}}(S) = (1/2)r_{\text{sym}}(\cup_{v \in S} U_v)$ for every $S \subseteq V$. Hence, the function f is half times the contraction of a symmetrized matroid rank function. Thus, if every symmetrized matroid rank function is α-hypergraph-approximable, then by Observation 5.1, the function f is also α-hypergraph-approximable. QED.

5.2 LOWER BOUND

In this section, we prove Theorem 5.1. In our first lemma, we show that it suffices to consider only hypergraphs with $\tilde{O}(n)$ hyperedges if we are willing to tolerate a constant loss in the approximation factor.

Lemma 5.1 (Few hyperedges suffice)**.** Let $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$ be a symmetric submodular function and $\beta \geq 1$ be a positive real number. Suppose that there exists a weighted hypergraph H whose cut function β-approximates f. Then, there exists a weighted hypergraph H' with $\tilde{O}(n)$ hyperedges whose cut function 2β-approximates f.

Proof. Applying Theorem 5.4 to H with $\epsilon = 1$ gives us that there exists a weighted hypergraph H' with $\tilde{O}(n)$ hyperedges whose cut function 2-approximates the cut function of H. Since the cut function of H β-approximates f, this means that the cut function of H' 2β-approximates f. QED.

Next we show that it suffices to restrict our attention to hypergraphs with rational hyperedge weights while again losing only a constant in the approximation factor (since we will be considering only hypergraphs with $O(n)$ hyperedges).

Lemma 5.2 (Bounded rational weights suffice)**.** Let $r : 2^V \to \mathbb{R}_+$ be a matroid rank function on ground set $V = [n]$. Suppose that there exists a hypergraph $H = (V, E)$ with hyperedge weights $w \colon E \to \mathbb{R}_+$ with $|E| = \tilde{O}(n)$ whose cut function $d : 2^{[n]} \to \mathbb{R}_{\geq 0}$ β-approximates r_{sym}

for some $\beta = o(n)$. Then, there exist hyperedge weights $w' \colon E \to \mathbb{Q}_+$ which assign to each hyperedge of H a positive rational weight p/q where $p, q \le n^3$ such that d' 2β-approximates r_{sym}, where $d' : 2^{[n]} \to \mathbb{R}_{\ge 0}$ is the cut function induced by the weight function w'.

Proof. Since r is the rank function of a matroid on ground set $V = [n]$, we have that $r(S) \le n$ for all $S \subseteq [n]$, and therefore $r_{\text{sym}}(S) \le n$ for all $S \subseteq [n]$. Consequently, if $w(e) > n$ for some $e \in E$ then we have $d(S) > n \ge r_{\text{sym}}(S)$ for some $S \subseteq [n]$, a contradiction. Thus, we conclude that $w(e) \le n$ for every $e \in E$.

We define the new weight function $w' : E \to \mathbb{R}_{\ge 0}$ by

$$w'(e) := \frac{\lfloor n^2 w(e) \rfloor}{n^2} \ \forall\ e \in E. \tag{5.11}$$

For every $e \in E$, the weight $w'(e)$ is a rational number p/q with $q = n^2$ and $p \le n^2 w(e) \le n^3$. Next, we show that the cut function $d' : 2^{[n]} \to \mathbb{R}_{\ge 0}$ induced by this weight function w' satisfies the required bounds for every subset $S \subseteq [n]$.

For every $e \in E$, we have that $w'(e) \le w(e)$. Thus, for every $S \subseteq [n]$, we have that $d'(S) \le d(S) \le r_{\text{sym}}(S)$. Moreover, for every $e \in E$, we have that $w'(e) \ge w(e) - 1/n^2$. Therefore, for every $S \subseteq [n]$, we have that

$$d'(S) \ge d(S) - |E|/n^2. \tag{5.12}$$

Let $S \subseteq [n]$. If $r_{\text{sym}}(S) = 0$, then $d'(S) \le d(S) = 0$, and so $r_{sym}(S) \le 2\beta d'(S)$. Suppose $r_{\text{sym}}(S) > 0$. Since $r_{\text{sym}}(S)$ is an integer, this means that $r_{\text{sym}}(S) \ge 1$, and therefore $1/\beta \le r_{\text{sym}}(S)/\beta \le d(S)$. Since $|E| = \tilde{O}(n)$, we have that $|E|/n^2 = \tilde{O}(\frac{1}{n})$, and since $\beta = o(n)$, we conclude that $|E|/n^2 < 1/2\beta \le d(S)/2$. Hence, Inequality (5.12) gives us that $d'(S) \ge d(S)/2$, and therefore, $r_{\text{sym}}(S) \le \beta d(S) \le 2\beta d'(S)$.

QED.

We will show the existence of our desired matroid rank function using the following theorem of Balcan and Harvey [91].

Theorem 5.6. [91] For every positive integer n and $k \ge 8$ with $k = 2^{o(n^{1/3})}$, there exists a family of sets $\mathcal{A} \subseteq 2^{[n]}$ and a family of matroids $\mathcal{M} = \{M_{\mathcal{B}} : \mathcal{B} \subseteq \mathcal{A}\}$ on the ground set $[n]$ with the following properties:

1. $|\mathcal{A}| = k$ and $|A| = \lfloor n^{1/3} \rfloor$ for every $A \in \mathcal{A}$.

2. For every $\mathcal{B} \subseteq \mathcal{A}$ and every $A \in \mathcal{A}$, we have

$$\text{rank}_{M_\mathcal{B}}(A) = \begin{cases} 8\lfloor \log k \rfloor & (\text{if } A \in \mathcal{B}) \\ |A| = \lfloor n^{1/3} \rfloor & (\text{if } A \in \mathcal{A} \setminus \mathcal{B}) \end{cases}.$$

3. For every $A_1, A_2 \in \mathcal{A}$ with $A_1 \neq A_2$, we have $|A_1 \cap A_2| \leq 4 \log k$.

4. For every $\mathcal{B} \subsetneq \mathcal{A}$, we have $\text{rank}_{M_\mathcal{B}}([n]) = \lfloor n^{1/3} \rfloor$.

We note that the version of the theorem given in [91] does not include the third and fourth properties. However, the proof for the variant of Theorem 5.6 with the first two properties given in [91] shows that the third and fourth properties also hold. We are now ready to prove Theorem 5.1. The following is a restatement of Theorem 5.1.

Theorem 5.7. For every sufficiently large positive integer n, there exists a symmetrized matroid rank function $r_{\text{sym}} : 2^{[n]} \to \mathbb{Z}_{\geq 0}$ on ground set $[n]$ such that r_{sym} is not α-hypergraph-approximable for $\alpha = o(n^{1/3}/\log^2 n)$.

Proof. For simplicity, we will assume that $n = 8^x$ for some positive integer x, so that $\log n$ and $n^{1/3}$ are both integers. If the theorem holds for n of this form, it holds for all sufficiently large n, since for any $8^x \leq n < 8^{x+1}$ we can extend a matroid M on ground set $[8^x]$ to a matroid M' on ground set $[n]$ which has the same independent sets.

For $k = n^{\log n}$, let $\mathcal{A}$ be a collection of subsets of $[n]$ and $\mathcal{M} = \{M_\mathcal{B} : \mathcal{B} \subseteq \mathcal{A}\}$ be the family of matroids on ground set $[n]$ with the properties guaranteed by Theorem 5.6. We note that $|\mathcal{M}| = 2^{n^{\log n}}$. For each $\mathcal{B} \subseteq \mathcal{A}$, let $r_{\text{sym}}^\mathcal{B}$ be the symmetrized rank function of $M_\mathcal{B}$ and let $\mathcal{F} := \{r_{\text{sym}}^\mathcal{B} : M_\mathcal{B} \in \mathcal{M}\}$ be the family of symmetrized rank functions of matroids in the family $\mathcal{M}$. We note that $\mathcal{F}$ is a family of $2^{n^{\log n}}$ symmetrized matroid rank functions over the ground set $[n]$. We will prove that there exists $r_{\text{sym}}^\mathcal{B} \in \mathcal{F}$ which is not α-hypergraph-approximable. Suppose for contradiction that for every function $r_{\text{sym}}^\mathcal{B} \in \mathcal{F}$ there exists a hypergraph $H_\mathcal{B}$ such that the cut function $d_\mathcal{B}$ of $H_\mathcal{B}$ satisfies $d_\mathcal{B}(S) \leq r_{\text{sym}}^\mathcal{B}(S) \leq \alpha(n) d_\mathcal{B}(S)$ for all $S \subseteq [n]$.

Let $r_{\text{sym}}^\mathcal{B} \in \mathcal{F}$. By Lemma 5.1, there exists a weighted hypergraph $H'_\mathcal{B}$ with $\tilde{O}(n)$ hyperedges such that its cut function $d' : 2^{[n]} \to \mathbb{R}_{\geq 0}$ satisfies

$$d'(S) \leq r_{\text{sym}}^\mathcal{B}(S) \leq 2\alpha d'(S) \; \forall \; S \subseteq [n]. \tag{5.13}$$

Applying Lemma 5.2 to the rank function $\text{rank}_\mathcal{B} : 2^{[n]} \to \mathbb{Z}_{\geq 0}$ of the matroid $M_\mathcal{B}$ and the hypergraph $H'_\mathcal{B}$ gives a hypergraph $H''_\mathcal{B}$ with $\tilde{O}(n)$ hyperedges all of whose weights are rational values p/q with $p, q \leq n^3$ such that the cut function $d'' : 2^{[n]} \to \mathbb{R}_{\geq 0}$ of $H''_{\mathcal{B}}$ satisfies

$$d''(S) \leq r^{\mathcal{B}}_{\text{sym}}(S) \leq 4\alpha d''(S) \ \forall\ S \subseteq [n]. \tag{5.14}$$

Let $\mathcal{H}$ be the family of weighted hypergraphs $\{H''_{\mathcal{B}} : r^{\mathcal{B}}_{\text{sym}} \in \mathcal{F}\}$.

We now count the number of weighted hypergraphs in $\mathcal{H}$. Each hypergraph in $\mathcal{H}$ has $\tilde{O}(n)$ hyperedges with each hyperedge having rational weight p/q where $p, q \leq n^3$. The number of potential hyperedges in a n-vertex hypergraph is $2^n - 1$, so for every $m \in \mathbb{Z}_+$ the number of simple n-vertex hypergraphs with m hyperedges is $\binom{2^n-1}{m} = O(2^{nm})$. Consequently, the number of possible simple hypergraphs with $\tilde{O}(n)$ hyperedges is $2^{\tilde{O}(n^2)}$. The number of positive rational numbers p/q with $p, q \in [n^3]$ is at most n^6, so the number of ways to assign a weight of this kind to each hyperedge of a hypergraph with $\tilde{O}(n)$ hyperedges is $n^{\tilde{O}(n)}$. Therefore the number of hypergraphs with $\tilde{O}(n)$ hyperedges each of which has a positive rational weight p/q where $p, q \in [n^3]$ is $2^{\tilde{O}(n^2)} n^{\tilde{O}(n)} = 2^{\tilde{O}(n^2)} = 2^{o(n^{\log n})}$. Hence, $|\mathcal{H}| = 2^{o(n^{\log n})}$.

Let $\mathcal{F}' := \{r^{\mathcal{B}}_{\text{sym}} \in \mathcal{F} \colon |B| \leq |\mathcal{A}| - 2\}$. Since $|\mathcal{F}| = 2^{n^{\log n}}$ and $|\mathcal{A}| = n^{\log n}$, we have that $|\mathcal{F}'| = \Omega(2^{n^{\log n}})$. Since $|\mathcal{F}'| = \Omega(2^{n^{\log n}})$ while $|\mathcal{H}| = 2^{o(n^{\log n})}$, there must exist two distinct functions $r^{\mathcal{B}_1}_{\text{sym}}, r^{\mathcal{B}_2}_{\text{sym}} \in \mathcal{F}'$ such that there is a single weighted hypergraph $H \in \mathcal{H}$ whose cut function $d : 2^{[n]} \to \mathbb{R}_{\geq 0}$ satisfies

$$d(S) \leq r^{\mathcal{B}_1}_{\text{sym}}(S) \leq 8\alpha d(S) \ \forall\ S \subseteq [n] \text{ and} \tag{5.15}$$

$$d(S) \leq r^{\mathcal{B}_2}_{\text{sym}}(S) \leq 8\alpha d(S) \ \forall\ S \subseteq [n]. \tag{5.16}$$

Since $\mathcal{B}_1 \neq \mathcal{B}_2$, at least one of $\mathcal{B}_1 \setminus \mathcal{B}_2$ and $\mathcal{B}_2 \setminus \mathcal{B}_1$ must be non-empty. We assume without loss of generality that $\mathcal{B}_1 \setminus \mathcal{B}_2 \neq \emptyset$. Let $S \in \mathcal{B}_1 \setminus \mathcal{B}_2$. By Theorem 5.6, we have that $\text{rank}_{M_{\mathcal{B}_1}}(S) = 8 \log^2 n$ and $\text{rank}_{M_{\mathcal{B}_2}}(S) = n^{1/3}$. Since $r^{\mathcal{B}_1}_{\text{sym}}, r^{\mathcal{B}_2}_{\text{sym}} \in \mathcal{F}'$, we have that $|\mathcal{B}_1|, |\mathcal{B}_2| \leq |\mathcal{A}| - 2$, and thus $|\mathcal{B}_1 \cup \{S\}|, |\mathcal{B}_2 \cup \{S\}| \leq |\mathcal{A}| - 1$. Therefore $\mathcal{A} \setminus (\mathcal{B}_1 \cup \{S\}), \mathcal{A} \setminus (\mathcal{B}_2 \cup \{S\}) \neq \emptyset$, so there exist sets $T_1 \in \mathcal{A} \setminus (\mathcal{B}_1 \cup \{S\}), T_2 \in \mathcal{A} \setminus (\mathcal{B}_2 \cup \{S\})$. By Theorem 5.6, we have that $\text{rank}_{M_{\mathcal{B}_1}}(T_1), \text{rank}_{M_{\mathcal{B}_2}}(T_2) = n^{1/3}$ and $|S \cap T_1|, |S \cap T_2| \leq 4 \log^2 n$. Therefore, $\text{rank}_{M_{\mathcal{B}_1}}(T_1 \setminus S), \text{rank}_{M_{\mathcal{B}_2}}(T_2 \setminus S) \geq n^{1/3} - 4 \log^2 n$, and so $\text{rank}_{M_{\mathcal{B}_1}}([n] \setminus S), \text{rank}_{M_{\mathcal{B}_2}}([n] \setminus S) \geq n^{1/3} - 4 \log^2 n$. Furthermore, since $T_1, T_2 \subseteq [n]$ we have that $\text{rank}_{M_{\mathcal{B}_1}}([n]), \text{rank}_{M_{\mathcal{B}_2}}([n]) \geq$

$n^{1/3}$, and so by Theorem 5.6, we have $\text{rank}_{M_{\mathcal{B}_1}}([n]), \text{rank}_{M_{\mathcal{B}_2}}([n]) = n^{1/3}$. Thus, we have that

$$r^{\mathcal{B}_1}_{\text{sym}}(S) = \text{rank}_{M_{\mathcal{B}_1}}(S) + \text{rank}_{M_{\mathcal{B}_1}}([n] \setminus S) - \text{rank}_{M_{\mathcal{B}_1}}([n]) \tag{5.17}$$

$$\leq 8\log^2 n + n^{1/3} - n^{1/3} = 8\log^2 n \quad \text{and} \tag{5.18}$$

$$r^{\mathcal{B}_2}_{\text{sym}}(S) = \text{rank}_{M_{\mathcal{B}_2}}(S) + \text{rank}_{M_{\mathcal{B}_2}}([n] \setminus S) - \text{rank}_{M_{\mathcal{B}_2}}([n]) \tag{5.19}$$

$$\geq n^{1/3} + (n^{1/3} - 4\log^2 n) - n^{1/3} = n^{1/3} - 4\log^2 n. \tag{5.20}$$

Therefore, by inequalities (5.15) and (5.16), we have that $d(S) \leq r^{\mathcal{B}_1}_{\text{sym}}(S) \leq 8\log^2 n$, and $8\alpha d(S) \geq r^{\mathcal{B}_2}_{\text{sym}}(S) \geq n^{1/3} - 4\log^2 n$. Hence, $\alpha = \Omega(n^{1/3}/\log^2 n)$. This contradicts the assumption that $\alpha = o(n^{1/3}/\log^2 n)$. QED.

5.3 UPPER BOUNDS

In this section, we show that certain subfamilies of symmetric submodular functions are constant-hypergraph-approximable. In particular, we show how to approximate concave linear functions and symmetrized rank functions of uniform and partition matroids using hypergraph cut functions. We recall that a set function $f : 2^V \to \mathbb{R}_{\geq 0}$ is a concave linear function if there exist weights $w : V \to \mathbb{R}_{\geq 0}$ and an increasing concave function $h : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that $f(S) = h(\sum_{v \in S} w_v)$. If all weights are one, then f_{sym} is symmetric submodular and moreover, the precise value of $f(S)$ depends only on the size $|S|$ and does not depend on the precise identify of the elements in S, so we call such functions f as anonymized concave linear functions. In Section 5.3.1, we consider the special case of anonymized concave linear functions and show that these are constant-hypergraph-approximable. We extend these ideas in Section 5.3.2 to show that symmetrized concave linear functions are constant-hypergraph-approximable. In Section 5.3.3 we use our result on approximating anonymized concave linear functions to show that the symmetrized rank functions of uniform and partition matroids are constant-hypergraph-approximable, proving Theorem 5.3.

5.3.1 Anonymized Concave Linear Functions

The main result of this section is Theorem 5.8, which states that anonymized concave linear functions are constant-hypergraph-approximable. We first show a few combinatorial inequalities that will be useful for our proof.

Claim 5.2. For every integer $n \geq 2$, $k \in \{1, \ldots, \frac{n}{2}\}$, and $r \in \{2, \ldots, n-k\}$, we have that

1. $\left(1 - \frac{r}{n-k}\right)^k \leq \frac{\binom{n-k}{r}}{\binom{n}{r}} \leq \left(1 - \frac{r}{n}\right)^k$.

2. $\left(1-\frac{k}{n-r}\right)^r \le \frac{\binom{n-k}{r}}{\binom{n}{r}}$

Proof. 1. We note that

$$\frac{\binom{n-k}{r}}{\binom{n}{r}} = \frac{(n-k)!/(r!(n-k-r!))}{n!/(r!(n-r)!)} \tag{5.21}$$

$$= \frac{(n-k)!}{n!} \cdot \frac{(n-r)!}{(n-k-r)!} \tag{5.22}$$

$$= \prod_{i=0}^{k-1} \frac{n-r-i}{n-i}. \tag{5.23}$$

We get the upper bound on $\binom{n-k}{r}/\binom{n}{r}$ by upper bounding every element of this product with $\frac{n-r}{n}$, and the lower bound by lower bounding every term of the product with $\frac{n-k-r}{n-k}$.

2. We note that

$$\frac{\binom{n-k}{r}}{\binom{n}{r}} = \frac{(n-k)!/(r!(n-k-r!))}{n!/(r!(n-r)!)} \tag{5.24}$$

$$= \frac{(n-k)!}{(n-k-r)!} \cdot \frac{(n-r)!}{n!} \tag{5.25}$$

$$= \prod_{i=0}^{r-1} \frac{n-k-i}{n-i}. \tag{5.26}$$

We obtain the lower bound by lower bounding every term of the product with $\frac{n-k-r}{n-r}$.

QED.

Claim 5.3. For every integer $n \ge 2$, $k \in \{1, \dots, \frac{n}{2}\}$, and $r \in \{2, \dots, n\}$, we have that

$$\frac{\binom{k}{r}}{\binom{n}{r}} \le \left(\frac{k}{n}\right)^r.$$

Proof. If $k < r$, the bound trivially holds, because $\binom{k}{r} = 0$. Otherwise, we have

$$\frac{\binom{k}{r}}{\binom{n}{r}} = \frac{k!/(r!(k-r)!)}{n!/(r!(n-r)!)} \tag{5.27}$$

$$= \frac{k!}{(k-r)!} \cdot \frac{(n-r)!}{n!} \tag{5.28}$$

$$= \prod_{i=0}^{r-1} \frac{k-i}{n-i}. \tag{5.29}$$

Upper bounding every term in the product with $\frac{k}{n}$ gives the desired bound. QED.

The following lemma is useful for proving the main theorem of this section.

Lemma 5.3. For every integer $n \geq 2$, $r \in \{2, \ldots, n\}$, and $X \subseteq [n]$ with $1 \leq |X| \leq \frac{n}{2}$, the set of hyperedges $\delta(X)$ that cross X in a complete r-uniform n-vertex hypergraph has the following size bound:

$$\frac{1}{4} \min\left\{\frac{|X|r}{n}, 1\right\} \leq \frac{|\delta(X)|}{\binom{n}{r}} \leq 4 \min\left\{\frac{|X|r}{n}, 1\right\}.$$

Proof. Let $k := |X|$. We note that the hyperedges which cross X are exactly those which are neither fully contained in X, nor fully contained in $V \setminus X$. Thus, the number of rank r hyperedges in $\delta(X)$ is exactly $\binom{n}{r} - \binom{n-k}{r} - \binom{k}{r}$.

Suppose $r > n - k$. Then, since $k \leq n/2$, we have that $r > k$ as well, so $|\delta(X)| = \binom{n}{r} - \binom{n-k}{r} - \binom{k}{r} = \binom{n}{r}$, and so we have $\frac{|\delta(X)|}{\binom{n}{r}} = 1$. Thus, we immediately have that $\frac{1}{4}\min\left\{\frac{|X|r}{n}, 1\right\} \leq \frac{|\delta(X)|}{\binom{n}{r}}$. Furthermore, we have that $kr > k(n-k) = kn - k^2$, so $\frac{kr}{n} > \frac{kn-k^2}{n} = k - \frac{k^2}{n} \geq k - \frac{k}{2} = \frac{k}{2}$, so we have that $\frac{|\delta(X)|}{\binom{n}{r}} \leq 4\min\left\{\frac{|X|r}{n}, 1\right\}$. Henceforth we assume $r \leq n - k$.

We case on the value of k.

- Case 1: $k \geq n/r$. Then $\min\left\{\frac{|X|r}{n}, 1\right\} = 1$. Since $\frac{|\delta(X)|}{\binom{n}{r}}$ is the fraction of the hyperedges which are in $\delta(X)$, it is trivially upper bounded by 1, and thus by $4\min\left\{\frac{kr}{n}, 1\right\}$. Therefore, it remains to show the lower bound. We have that

$$\frac{|\delta(X)|}{\binom{n}{r}} = \frac{\binom{n}{r} - \binom{n-k}{r} - \binom{k}{r}}{\binom{n}{r}} \tag{5.30}$$

$$\geq 1 - \left(1 - \frac{r}{n}\right)^k - \left(\frac{k}{n}\right)^r \tag{5.31}$$

$$\geq 1 - e^{-kr/n} - \left(\frac{k}{n}\right)^r \tag{5.32}$$

$$\geq 1 - \frac{1}{e} - \frac{1}{4} \tag{5.33}$$

$$\geq \frac{1}{4} \tag{5.34}$$

$$= 0.25 \min\left\{\frac{kr}{n}, 1\right\}. \tag{5.35}$$

Here the second line follows from the upper bound in the first conclusion of Claim 5.2

and the upper bound in Claim 5.3, and the fourth follows from our assumptions that $n/r \leq k \leq n/2$ and $r \geq 2$.

- Case 2: $k < n/r$. Then $\min\left\{\frac{|X|r}{n}, 1\right\} = \frac{kr}{n}$. Once again, we need to show a lower bound and an upper bound. We begin with the lower bound:

$$\frac{|\delta(X)|}{\binom{n}{r}} = \frac{\binom{n}{r} - \binom{n-k}{r} - \binom{k}{r}}{\binom{n}{r}} \tag{5.36}$$

$$\geq 1 - \left(1 - \frac{r}{n}\right)^k - \left(\frac{k}{n}\right)^r \tag{5.37}$$

$$\geq 1 - e^{-kr/n} - \left(\frac{k}{n}\right)^r \tag{5.38}$$

$$\geq 1 - \left(1 - \frac{kr}{2n}\right) - \left(\frac{k}{n}\right)^r \tag{5.39}$$

$$= \frac{kr}{2n} - \left(\frac{k}{n}\right)^r \tag{5.40}$$

$$\geq \frac{kr}{2n} - \left(\frac{kr}{2n}\right)^2 \tag{5.41}$$

$$= \frac{kr}{2n}\left(1 - \frac{kr}{2n}\right) \tag{5.42}$$

$$\geq \frac{kr}{4n}. \tag{5.43}$$

Here the second line follows from the upper bound in the first conclusion of Claim 5.2 and the upper bound in Claim 5.3, the fourth from the Taylor expansion of e^x, the sixth from the fact that $r \geq 2$, and the last line from the assumption that $k < n/r$.

Now we show the upper bound. Since the total number of hyperedges in the graph is $\binom{n}{r}$, we have that $|\delta(X)| \leq \binom{n}{r}$. Hence, $|\delta(X)|/\binom{n}{r} \leq 1$. It remains to show that $|\delta(X)|/\binom{n}{r} \leq 4|X|r/n = 4rk/n$. We consider 3 subcases based on the values of r and k:

  - Subcase 1: $r \geq n/4$. Since $r \geq n/4$ and $|X| \geq 1$, we have that $\frac{|X|r}{n} \geq \frac{1}{4}$. Therefore

$$\frac{|\delta(X)|}{\binom{n}{r}} \leq 1 \leq 4\frac{|X|r}{n} \tag{5.44}$$

– Subcase 2: $r < n/4$ and $k < r$. In this case, we have that $\binom{k}{r} = 0$. Therefore,

$$\begin{aligned}
\frac{|\delta(X)|}{\binom{n}{r}} &= \frac{\binom{n}{r} - \binom{n-k}{r} - \binom{k}{r}}{\binom{n}{r}} && (5.45)\\
&= \frac{\binom{n}{r} - \binom{n-k}{r}}{\binom{n}{r}} && (5.46)\\
&= 1 - \frac{\binom{n-k}{r}}{\binom{n}{r}} && (5.47)\\
&\le 1 - \left(1 - \frac{r}{n-k}\right)^k && (5.48)\\
&\le 1 - e^{-2rk/(n-k)} && (5.49)\\
&\le 1 - \left(1 - \frac{2rk}{n-k}\right) && (5.50)\\
&= \frac{2rk}{n-k} && (5.51)\\
&\le \frac{4rk}{n}. && (5.52)
\end{aligned}$$

The fourth line follows from the lower bound in the first conclusion of Claim 5.2. The fifth line follows from observing that $0 < k/(n-r) \le 2/3$ (since $k \le n/2$ and $r \le n/4$) and $\ln(1-x) \ge -2x$ for every $x \in (0, 2/3]$. The sixth line follows from the Taylor expansion of e^x, and the last line follows from the fact that $k \le n/2$.

– Subcase 3: $r < n/4$ and $k \geq r$. In this case we have that

$$\begin{aligned}
\frac{|\delta(X)|}{\binom{n}{r}} &= \frac{\binom{n}{r} - \binom{n-k}{r} - \binom{k}{r}}{\binom{n}{r}} && (5.53)\\
&\leq \frac{\binom{n}{r} - \binom{n-k}{r}}{\binom{n}{r}} && (5.54)\\
&= 1 - \frac{\binom{n-k}{r}}{\binom{n}{r}} && (5.55)\\
&\leq 1 - \left(1 - \frac{k}{n-r}\right)^r && (5.56)\\
&\leq 1 - e^{-2rk/(n-r)} && (5.57)\\
&\leq 1 - \left(1 - \frac{2rk}{n-r}\right) && (5.58)\\
&= \frac{2rk}{n-r} && (5.59)\\
&\leq \frac{2rk}{3n/4} && (5.60)\\
&\leq \frac{4rk}{n}. && (5.61)
\end{aligned}$$

The fourth line follows from the lower bound in the second conclusion of Claim 5.2. The fifth line follows from observing that $0 < k/(n-r) \leq 2/3$ (since $k \leq n/2$ and $r \leq n/4$) and $\ln(1-x) \geq -2x$ for every $x \in (0, 2/3]$. The sixth line follows from the Taylor expansion of e^x. The second to last line follows from the fact that $r < n/4$.

QED.

The following is the main theorem of this section.

Theorem 5.8. Let n be a positive real number and $h\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be a function such that h is concave on $[0, n]$ and $h(x) = h(n-x)$ for every $x \in [0, n]$. Then, the symmetric submodular function $f : 2^V \to \mathbb{R}_{\geq 0}$ over the ground set $V = [n]$ defined by $f(S) := h(|S|)\ \forall\ S \subseteq V$ is 64-hypergraph-approximable.

Proof. To simplify our notation, we define $a_x := h(x) - h(x-1)$ for $x \in \{1, \ldots, \lceil n/2 \rceil\}$. A hypergraph is uniform if all its hyperedges have the same size and a complete t-uniform hypergraph consists of all hyperedges of size t. We define H as the union of $\lceil n/2 \rceil$ different hypergraphs, $G_0, \ldots, G_{\lceil n/2 \rceil}$, each of which is a uniform hypergraph over the vertex set V and each of whose hyperedges are weighted uniformly. Formally, H is the union of:

1. A complete $\lceil \frac{n}{x} \rceil$-uniform hypergraph G_x, with a total weight of $(a_x - a_{x+1})(x/8)$ equally distributed among its hyperedges for each $x \in \{1, \ldots, \lceil n/2 \rceil - 1\}$, i.e., $w(e) = (a_x - a_{x+1})(x/8)/\binom{n}{\lceil \frac{n}{x} \rceil}$ for every hyperedge $e \in E(G_x)$ (we note that $a_x - a_{x+1} \geq 0$ since h is concave).

2. A complete 2-uniform hypergraph $G_{\lceil n/2 \rceil}$, with a total weight of $a_{\lceil n/2 \rceil}(n/32)$ equally distributed among its hyperedges.

3. A hypergraph G_0 consisting of a single n-vertex hyperedge of weight $h(0)/64$.

Let d be the cut function of the hypergraph H we have just defined. In order to show that d 64-approximates f, we will consider an arbitrary subset C of size k and bound its cut value in H. Since we know that d and f are both symmetric, we assume without loss of generality that $1 \leq k \leq n/2$.

We now compute the weight of hyperedges crossing C in H. We recall that $|C| = k \leq n/2$. We begin with the easy cases. $\delta(C)$ will certainly cut the single hyperedge of G_0 for a weight of exactly $h(0)/64$. The hyperedges in $G_{\lceil n/2 \rceil}$ have rank 2. Therefore, by Lemma 5.3, the number of hyperedges crossing C in $G_{\lceil n/2 \rceil}$ is at least a $\frac{k}{2n}$ fraction and at most a $\frac{8k}{n}$ fraction of the hyperedges in $G_{\lceil n/2 \rceil}$, for a total weight between $a_{\lceil n/2 \rceil}k/64$ and $a_{\lceil n/2 \rceil}k/4$.

Next, we compute the weight of hyperedges crossing C in $G_1, \ldots, G_k$. Let us consider G_x for a fixed $x \in \{1, \ldots, k\}$. Let $r := \lceil \frac{n}{x} \rceil$. We have that $r \geq \frac{n}{x} \geq \frac{n}{k}$, so $\frac{kr}{n} \geq 1$. Therefore, by Lemma 5.3, the number of hyperedges crossing C in G_x is at least a quarter of the hyperedges of G_x. We also know that even if all hyperedges in G_x cross C, the weight of those hyperedges is only $(a_x - a_{x+1})(x/8)$. Therefore, the weight of hyperedges crossing C in G_x is between $(a_x - a_{x+1})(x/32)$ and $(a_x - a_{x+1})(x/8)$.

Next, we compute the weight of hyperedges crossing C in $G_{k+1}, \ldots, G_{\lceil n/2 \rceil - 1}$. Let us consider G_x for a fixed $x \in \{k+1, \ldots, \lceil \frac{n}{2} \rceil - 1\}$. Let $r := \lceil \frac{n}{x} \rceil$. Then, $2 \leq r \leq \frac{2n}{x} < \frac{2n}{k}$. Therefore, $\frac{kr}{n} \leq 2$, and hence,

$$\frac{k}{2x} \leq \frac{kr}{2n} \leq \min\left(\frac{kr}{n}, 1\right) \leq \frac{kr}{n} \leq \frac{2k}{x}. \tag{5.62}$$

From these inequalities and Lemma 5.3, we conclude that the number of hyperedges crossing C in G_x is at least a $\frac{k}{8x}$ fraction and at most a $\frac{8k}{x}$ fraction of hyperedges of G_x. Therefore, the weight of hyperedges crossing C in G_x is at least $(a_x - a_{x+1})k/64$ and at most $(a_x - a_{x+1})k$.

Therefore, if $d(C)$ is the weight of hyperedges crossing C in H, then

$$\frac{1}{64}\left(h(0) + a_{\lceil n/2\rceil}k + \sum_{x=1}^{k}(a_x - a_{x+1})x + \sum_{x=k+1}^{\lceil n/2\rceil-1}(a_x - a_{x+1})k\right) \tag{5.63}$$

$$\leq \frac{1}{64}\left(h(0) + a_{\lceil n/2\rceil}k + \sum_{x=1}^{k}2(a_x - a_{x+1})x + \sum_{x=k+1}^{\lceil n/2\rceil-1}(a_x - a_{x+1})k\right) \tag{5.64}$$

$$\leq d(C) \tag{5.65}$$

$$\leq \frac{h(0)}{64} + a_{\lceil n/2\rceil}\frac{k}{4} + \sum_{x=1}^{k}(a_x - a_{x+1})\frac{x}{8} + \sum_{x=k+1}^{\lceil n/2\rceil-1}(a_x - a_{x+1})k \tag{5.66}$$

$$\leq h(0) + a_{\lceil n/2\rceil}k + \sum_{x=1}^{k}(a_x - a_{x+1})x + \sum_{x=k+1}^{\lceil n/2\rceil-1}(a_x - a_{x+1})k. \tag{5.67}$$

Here, expression (5.67) is 64 times expression (5.63), so our proof is complete if we can show that expression (5.67) evaluates to $h(k)$ (recall that $h(k) = f(C)$. The next claim completes the proof by showing this. QED.

Claim 5.4. For every $k \in \{0, 1, 2, \ldots, \lceil n/2\rceil\}$, we have that

$$h(k) = h(0) + a_{\lceil n/2\rceil}k + \sum_{x=1}^{k}(a_x - a_{x+1})x + \sum_{x=k+1}^{\lceil n/2\rceil-1}(a_x - a_{x+1})k$$

Proof. To show this, we simplify the two summations appearing on the RHS. The second summation telescopes to yield $(a_{k+1} - a_{\lceil n/2\rceil})k$. To simplify the first summation, we note that $\sum_{x=1}^{j}(a_x - a_{x+1})x = \sum_{x=1}^{j}(2h(x) - h(x+1) - h(x-1))x$. For every x from 1 to $j-1$, $h(x)$ is added $2x$ times and subtracted $2x$ times in this summation, so it does not contribute at all. Therefore, we conclude that $\sum_{x=1}^{j}(a_x - a_{x+1})x = (j+1)h(x) - jh(x+1) - h(0)$.

Using the simplifications we have derived for each of the summations on the RHS, we find that the RHS is

$$h(0) + a_{\lceil n/2\rceil}k + ((k+1)h(k) - kh(k+1) - h(0)) + (a_{k+1} - a_{\lceil n/2\rceil})k \tag{5.68}$$

$$= ka_{k+1} + (k+1)h(k) - kh(k+1) \tag{5.69}$$

$$= k(h(k+1) - h(k)) + (k+1)h(k) - kh(k+1) \tag{5.70}$$

$$= h(k). \tag{5.71}$$

QED.

5.3.2 Symmetrized Concave Linear Functions

In this section, we prove Theorem 5.2—i.e., symmetrized concave linear functions are constant-hypergraph-approximable. Our approach is to first construct a hypergraph on a much larger vertex set than the ground set V, using the result of Theorem 5.8, and then contract subsets of the vertices of this hypergraph to obtain a hypergraph on the vertex set V with the desired property.

Theorem 5.9. Let V be a ground set, $w\colon V \to \mathbb{R}_+$, and $h\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be an increasing concave function. Then, the symmetric submodular function $f\colon 2^V \to \mathbb{R}_+$ defined by

$$f(S) := h\left(\sum_{v\in S} w(v)\right) + h\left(\sum_{v\in V\setminus S} w(v)\right) - h\left(\sum_{v\in V} w(v)\right) - h(0)\ \forall\ S \subseteq V$$

is 128-hypergraph-approximable.

Proof. Let $n := |V|$. For ease of notation, we will use $w(S) := \sum_{v\in S} w(v)$ for all $S \subseteq V$. Let $g\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be defined by $g(x) := h(x)+h(w(V)-x)-h(w(V))-h(0)$ for all $x \geq 0$. Then $f(S) = g(w(S))$. Since h is concave, and $h(w(V)-x)$ is $h(x)$ reflected over a vertical line at $x = w(V)/2$, the function $h(w(V)-x)$ is also concave. We also note that $-h(w(V))-h(0)$ is a constant, and constant functions are concave. Therefore, g is a sum of concave functions, and hence, g is concave as well. Since g is concave, it is also continuous. Therefore, for every $x \in \mathbb{R}_+$ such that $g(x) \neq 0$, there exists a positive real number ε_x such that for every real number y with $x - \varepsilon_x < y < x + \varepsilon_x$, we have $g(x)/\sqrt{2} \leq g(y) \leq \sqrt{2}g(x)$. Let $\varepsilon_{min} = \min\{\varepsilon_{w(S)} \colon \emptyset \neq S \subset V\}$. Let $q := \lceil 2nw(V)/\varepsilon_{min}\rceil$. We note that $w(V)/q \leq \varepsilon_{w(S)}/2n$ for every $S \subset V$.

Claim 5.5. There exist positive integers p_v for each $v \in V$ such that:

1. For every $v \in V$, we have that $w(v) - \frac{\varepsilon_{min}}{n} < \frac{p_v w(V)}{q} < w(v) + \frac{\varepsilon_{min}}{n}$.
2. $\sum_{v\in V} p_v = q$.

Proof. By our choice of q, for every $v \in V$, we have that $w(v) - w(V)/q > w(v) - \frac{\varepsilon_{min}}{n}$. Therefore, for each $v \in V$ we can choose a positive integer p_v such that $w(v) - \frac{\varepsilon_{min}}{n} < p_v w(V)/q \leq w(v)$, and thus we can choose a collection of integers p_v which satisfies the first condition of the claim as well as $\sum_{v\in V} p_v \leq q$.

Consider a collection of positive integers p_v for each $v \in V$ which maximizes $\sum_{v\in V} p_v$ subject to satisfying the first condition of the claim and the inequality $\sum_{v\in V} p_v \leq q$. Suppose for contradiction that these integers do not satisfy the second condition of the claim. Then,

$\sum_{v\in V} p_v < q$, so $\sum_{v\in V} p_v w(V)/q < w(V)$, so there must exist some $u \in V$ for which $p_u w(V)/q < w(u)$. By our choice of q, we have that

$$\frac{(p_u+1)w(V)}{q} = \frac{p_u w(V)}{q} + \frac{w(V)}{q} < w(u) + \frac{w(V)}{q} \le w(u) + \frac{\varepsilon_{min}}{2n} < w(u) + \frac{\varepsilon_{min}}{n}. \quad (5.72)$$

Thus, we can increase p_u by 1 while still satisfying the first condition of the claim. Also, since $\sum_{v\in V} p_v < q$, we have that $1 + \sum_{v\in V} p_v \le q$, so we can increase p_u by 1 while maintaining that the sum of all the integers p_v is at most q. This contradicts our assumption that the integers p_v maximized $\sum_{v\in V} p_v$ subject to satisfying the first constraint of the claim and the inequality $\sum_{v\in V} p_v \le q$. Thus, a collection of positive integers satisfying the conditions of the claim exists. QED.

Choose a positive integer p_v for each $v \in V$ such that the chosen integers satisfy the conditions of Claim 5.5. For each $v \in V$, we create a set U_v containing p_v new vertices, and we define $U := \bigcup_{v\in V} U_v$. We note that $|U| = \sum_{v\in V} p_v = q$. We define functions $h_1 \colon \mathbb{R}_{\ge 0} \to \mathbb{R}_{\ge 0}$, $f_1 \colon [0,q] \to \mathbb{R}_+$, and $f_2 \colon [0,q] \to \mathbb{R}_+$ by $h_1(x) = h(xw(V)/q)$, $f_1(x) = h_1(x) + h_1(q-x) - h_1(q) - h_1(0)$, and $f_2(x) = f_1(x)/\sqrt{2}$. We note that h_1 is concave since it is a rescaling of h by a constant factor, and $h_1(q-x)$ is concave, since it is h_1 reflected over the vertical line at $x = q/2$. Thus, f_1 is the sum of concave functions, and hence, f_1 is concave. Finally, f_2 is a constant multiple of a concave function, so f_2 is concave as well. Furthermore, by definition, $f_2(q-x) = f_2(x)$. Applying Theorem 5.8 to q and f_2, we conclude that there exists a hypergraph H' with vertex set U whose cut function d' satisfies

$$d'(S) \le f_1(|S|)/\sqrt{2} \le 64 d'(S) \ \forall\ S \subseteq U, \quad (5.73)$$

Let H be the hypergraph obtained from H' by contracting each set U_v of vertices into a vertex $v \in V$. Let d be the cut function of H. To complete the proof, we will show that $d(S) \le f(S) \le 128 d(S)$ for every $S \subseteq V$. We first consider the special cases of $S = \emptyset$ and $S = V$. For both these cases, we have that $f(S) = 0 = d(S)$ by definition. Next, let us consider an arbitrary non-empty set $S \subset V$. Let $U_S := \bigcup_{v\in S} U_v$ be the corresponding set of vertices in H'. We note that by construction of H, we have that $d(S) = d'(U_S)$. Therefore,

$$d(S) \le f_1(|U_S|)/\sqrt{2} \le 64 d(S). \quad (5.74)$$

We note that

$$|U_S| = \sum_{v\in S} |U_v| = \sum_{v\in S} p_v. \quad (5.75)$$

Therefore, by definition,

$$
\begin{aligned}
f_1(|U_S|) &= f_1\left(\sum_{v\in S} p_v\right) && (5.76)\\
&= h_1\left(\sum_{v\in S} p_v\right) + h_1\left(q - \sum_{v\in S} p_v\right) - h_1(q) - h_1(0) && (5.77)\\
&= h\left(\sum_{v\in S} \frac{p_v w(V)}{q}\right) + h\left(w(V) - \sum_{v\in S} \frac{p_v w(V)}{q}\right) - h(w(V)) - h(0) && (5.78)\\
&= g\left(\sum_{v\in S} \frac{p_v w(V)}{q}\right). && (5.79)
\end{aligned}
$$

For each $v \in S$, we have that $w(v) - \varepsilon_{min}/n < p_v w(V)/q < w(v) + \varepsilon_{min}/n$. We also have that $|S| \le n$. Therefore,

$$
\begin{aligned}
w(S) - \varepsilon_{w(S)} &\le w(S) - \varepsilon_{min} && (5.80)\\
&\le w(S) - \frac{|S|\varepsilon_{min}}{n} && (5.81)\\
&< \sum_{v\in S} \frac{p_v w(V)}{q} && (5.82)\\
&< w(S) + \frac{|S|\varepsilon_{min}}{n} && (5.83)\\
&\le w(S) + \varepsilon_{min} && (5.84)\\
&\le w(S) + \varepsilon_{w(S)}. && (5.85)
\end{aligned}
$$

So by definition of $\varepsilon_{w(S)}$, we have that

$$
\frac{f(S)}{\sqrt{2}} = \frac{g(w(S))}{\sqrt{2}} \le g\left(\sum_{v\in S} \frac{p_v w(V)}{q}\right) \le \sqrt{2} g(w(S)) = \sqrt{2} f(S). \qquad (5.86)
$$

Thus $f(S)/\sqrt{2} \le f_1(|U_S|) \le \sqrt{2} f(S)$, and so by inequality (5.74) we have that

$$
d(S) \le f_1(|U_S|)/\sqrt{2} \le f(S) \le \sqrt{2} f_1(|U_S|) \le 128 d(S). \qquad (5.87)
$$

QED.

5.3.3 Symmetrized Matroid Rank Functions

In this section, we prove Theorem 5.3 which states that symmetrized rank function of uniform and partition matroids are constant-hypergraph-approximable (see Section 5.1.2 for definitions of uniform and partition matroids). We begin with uniform matroids.

Lemma 5.4. The symmetrized rank function of a uniform matroid is 64-hypergraph-approximable.

Proof. Let $r : 2^V \to \mathbb{R}_{\geq 0}$ be the rank function of the uniform matroid on ground set V with budget k and $r_{\text{sym}} : 2^V \to \mathbb{R}_{\geq 0}$ be the symmetrized rank function. We note that $r(S) = \min\{|S|, k\}$ for every $S \subseteq V$. If $k > |V|$, then $r_{\text{sym}}(S) = 0$ for every $S \subseteq V$ and hence, r_{sym} is 1-hypergraph-approximable using the empty hypergraph. So, we may assume that $k \leq |V|$. Then, for every $S \subseteq V$, we have that

$$\begin{aligned} r_{\text{sym}}(S) &= r(S) + r(V \setminus S) - r(V) && (5.88) \\ &= \min\{|S|, k\} + \min\{|V \setminus S|, k\} - \min\{|V|, k\} && (5.89) \\ &= \min\{|S|, |V \setminus S|, k, |V| - k\}. && (5.90) \end{aligned}$$

Let $n := |V|$ and consider the function $h : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ defined by

$$h(x) = \min\{x, n - x, k, n - k\}. \tag{5.91}$$

Then, h is concave on $[0, n]$ and $h(x) = h(n - x)$ for every $x \in [0, n]$ and $r_{\text{sym}}(S) = h(|S|)$ for every $S \subseteq V$. Therefore, by Theorem 5.8, we have that r_{sym} is 64-hypergraph-approximable. QED.

Next, we show that the symmetrized rank functions of partition matroids are constant-hypergraph-approximable.

Theorem 5.10. The symmetrized rank function of a partition matroid is 64-hypergraph-approximable.

Proof. Let $\mathcal{M} = (V, \mathcal{I})$ be a partition matroid on ground set V with rank function $r : 2^V \to \mathbb{Z}_{\geq 0}$ that is associated with the partition $V_1, \ldots, V_t$ of the ground set V and budgets $b_1, \ldots, b_t \in \mathbb{Z}_{\geq 0}$. For $i \in [t]$, we define a function $f_i \colon 2^{V_i} \to \mathbb{Z}_{\geq 0}$ by $f_i(S) := r_i(S) + r_i(V_i \setminus S) - r_i(V_i)$ where r_i is the rank function of the uniform matroid on ground set V_i with budget b_i. Then, the symmetrized rank function of the partition matroid $\mathcal{M}$ can be written as $r_{\text{sym}}(S) = \sum_{i=1}^{t} f_i(S \cap V_i)$. Moreover, each f_i is the symmetrized rank function of a uniform matroid. By Lemma 5.4, for each $i \in [t]$, there exists a weighted hypergraph G_i with cut function d_i such that

$$d_i(S) \leq f_i(S) \leq 64 d_i(S) \ \forall \ S \subseteq P_i. \tag{5.92}$$

Let G be the hypergraph on V formed by taking the union of the hypergraphs G_i for each $i \in [t]$. Since the vertex sets of the hypergraphs G_i are pairwise disjoint, the cut function $d : 2^V \to \mathbb{R}_{\geq 0}$ of G satisfies $d(S) = \sum_{i=1}^{t} d_i(S \cap V_i)$, and therefore G is a weighted hypergraph which fulfills the requirements of the theorem. QED.

5.4 CONCLUSION AND OPEN PROBLEMS

In this chapter, we investigated the approximability of symmetric submodular functions using hypergraph cut functions. We proved that it suffices to understand the approximability of symmetrized matroid rank functions.

On the upper bound side, we showed that symmetrized concave linear functions and symmetrized rank functions of uniform and partition matroids are constant-approximable using hypergraph cut functions. Our upper bounds for uniform and partition matroids raise the question of whether symmetrized rank functions of constant-depth laminar matroids are constant-approximable using hypergraph cut functions. Laminar matroids generalize partition matroids. A family $\mathcal{L}$ of subsets is said to be laminar if for every pair of sets $A, B \in \mathcal{L}$, we have that either $A \subseteq B$ or $B \subseteq A$ or $A \cap B = \emptyset$. Suppose that we have a laminar family $\mathcal{L}$ and a function $b : \mathcal{L} \to \mathbb{Z}_{\geq 0}$. Then, the independent sets of the laminar matroid associated with $\mathcal{L}$ and b are those subsets $A \subseteq V$ for which $|A \cap S| \leq b(S)$ for every $S \in \mathcal{L}$. The laminar matroid is said to have depth d, if the longest nested sequence of sets in the laminar family $\mathcal{L}$ has exactly d sets.

On the lower bound side, we showed that there exist symmetrized matroid rank functions on n-element ground sets that cannot be $o(n^{1/3}/\log^2 n)$-approximated using hypergraph cut functions, thus ruling out constant-approximability of symmetric submodular functions using hypergraph cut functions. Our results raise the natural open question of whether every symmetric submodular function on n-element ground set is $O(\sqrt{n})$-hypergraph approximable.

REFERENCES

[1] T. E. Harris and F. S. Ross, “Fundamentals of a method for evaluating rail net capacities,” Rand Corp. Santa Monica, CA, Tech. Rep., 1955.

[2] L. R. Ford and D. R. Fulkerson, “Maximal flow through a network,” *Canadian journal of Mathematics*, vol. 8, pp. 399–404, 1956.

[3] P. Elias, A. Feinstein, and C. Shannon, “A note on the maximum flow through a network,” *IRE Transactions on Information Theory*, vol. 2, no. 4, pp. 117–119, 1956.

[4] G. Dantzig and D. R. Fulkerson, “On the max flow min cut theorem of networks,” in *The Basic George B. Dantzig.* Stanford University Press, 2003, pp. 225–231.

[5] L. R. Ford and D. R. Fulkerson, *Flows in networks.* Princeton University Press, 1962.

[6] R. E. Gomory and T. C. Hu, “Multi-terminal network flows,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 9, no. 4, pp. 551–570, 1961.

[7] J. Li and D. Panigrahi, “Deterministic min-cut in poly-logarithmic max-flows,” in *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS, 2020, pp. 85–92.

[8] C. J. Colbourn, *The combinatorics of network reliability.*, ser. International Series of Monographs on Computer Science. Oxford University Press, 1987.

[9] R. A. Botafogo, “Cluster analysis for hypertext systems,” in *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR, 1993, pp. 116–125.

[10] Y. Boykov and V. Kolmogorov, “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[11] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, 2006.

[12] S. Mukhopadhyay and D. Nanongkai, “Weighted min-cut: sequential, cut-query, and streaming algorithms,” in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC, 2020, pp. 496–509.

[13] P. Gawrychowski, S. Mozes, and O. Weimann, “Minimum Cut in $O(m \log^2 n)$ Time,” in *Proceedings of the 47th International Colloquium on Automata, Languages and Programming*, ser. ICALP, 2020, pp. 57:1–57:15.

[14] J. Li, “Deterministic mincut in almost-linear time,” in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC, 2021, pp. 384–395.

[15] E. Dinits, A. Karzanov, and M. Lomonosov, "On the structure of a family of minimal weighted cuts in a graph," *Studies in Discrete Optimization (in Russian)*, pp. 290–306, 1976.

[16] D. Karger, "Global Min-Cuts in RNC, and Other Ramifications of a Simple Min-Cut Algorithm," in *Proceedings of the 4th annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA, 1993, p. 21–30.

[17] K. J. Ahn and S. Guha, "Graph sparsification in the semi-streaming model," in *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part II*, ser. ICALP, 2009, pp. 328–338.

[18] K. J. Ahn, S. Guha, and A. McGregor, "Analyzing graph structure via linear measurements," in *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA, 2012, pp. 459–467.

[19] K. J. Ahn, S. Guha, and A. McGregor, "Graph sketches: Sparsification, spanners, and subgraphs," in *Proceedings of the 31st Symposium on Principles of Database Systems*, ser. PODS, 2012, pp. 5–14.

[20] D. Kogan and R. Krauthgamer, "Sketching cuts in graphs and hypergraphs," in *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ser. ITCS, 2015, pp. 367–376.

[21] D. Karger, "Minimum cuts in near-linear time," *Journal of the ACM (JACM)*, vol. 47, no. 1, pp. 46–76, 2000, announced at STOC'96.

[22] A. A. Benczúr, "A representation of cuts within 6/5 times the edge connectivity with applications," in *Proceedings of the 36th Annual IEEE Foundations of Computer Science*, ser. FOCS, 1995, pp. 92–102.

[23] A. A. Benczúr, "Cut structures and randomized algorithms in edge-connectivity problems," Ph.D. dissertation, Massachusetts Institute of Technology, 1997.

[24] A. A. Benczúr and M. Goemans, "Deformable polygon representation and near-mincuts," in *Building Bridges.* Springer, 2008, pp. 103–135.

[25] A. Karlin, N. Klein, and S. O. Gharan, "A (slightly) improved approximation algorithm for metric tsp," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC, 2021, p. 32–45.

[26] C. Alpert and A. Kahng, "Recent developments in netlist partitioning: A survey," *Integration: the VLSI Journal*, vol. 19, no. 1-2, pp. 1–81, 1995.

[27] L. Zhao, "Approximation algorithms for partition and design problems in networks," Ph.D. dissertation, Graduate School of Informatics, Kyoto University, Japan, 2002.

[28] S. Tan, J. Bu, C. Chen, B. Xu, C. Wang, and X. He, "Using rich social media information for music recommendation via hypergraph model," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 7, no. 1, pp. 1–22, 2011.

[29] J. Zhu, J. Zhu, S. Ghosh, W. Wu, and J. Yuan, "Social influence maximization in hypergraph in social networks," *IEEE Transactions on Network Science and Engineering*, vol. 6, no. 4, pp. 801–811, 2018.

[30] U. Feige, J. H. Kim, and E. Ofek, "Witnesses for non-satisfiability of dense random 3cnf formulas," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS, 2006, pp. 497–508.

[31] R. Klimmek and F. Wagner, "A simple hypergraph min cut algorithm," Institute Of Computer Science, Freie Universitat, Tech. Rep. B 96-02, 1996.

[32] W.-K. Mak and M. D. F. Wong, "A fast hypergraph min-cut algorithm for circuit partitioning," *Integration: the VLSI Journal*, vol. 30, no. 1, pp. 1–11, 2000.

[33] M. Queyranne, "Minimizing symmetric submodular functions," *Mathematical Programming*, vol. 82, no. 1-2, pp. 3–12, 1998.

[34] M. Ghaffari, D. Karger, and D. Panigrahi, "Random contractions and sampling for hypergraph and hedge connectivity," in *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA, 2017, pp. 1101–1114.

[35] K. Chandrasekaran, C. Xu, and X. Yu, "Hypergraph k-cut in randomized polynomial time," *Mathematical Programming (Preliminary version in SODA 2018)*, vol. 186, pp. 85–113, 2019.

[36] K. Fox, D. Panigrahi, and F. Zhang, "Minimum cut and minimum k-cut in hypergraphs via branching contractions," in *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA, 2019, pp. 881–896.

[37] E. Cheng, "Edge-augmentation of hypergraphs," *Mathematical Programming*, vol. 84, pp. 443–465, 1999.

[38] C. Chekuri and C. Xu, "Minimum cuts and sparsification in hypergraphs," *SIAM Journal on Computing (Preliminary version in SODA 2017)*, vol. 47, no. 6, pp. 2118–2156, 2018.

[39] D. M. Topkis, *Supermodularity and complementarity.* Princeton University Press, 2011.

[40] L. S. Shapley, "Cores of convex games," *International journal of game theory*, vol. 1, no. 1, pp. 11–26, 1971.

[41] W. Hanson and R. K. Martin, "Optimal bundle pricing," *Management Science*, vol. 36, no. 2, pp. 155–174, 1990.

[42] S. Fujishige, "Polymatroidal dependence structure of a set of random variables," *Information and control*, vol. 39, no. 1, pp. 55–72, 1978.

[43] A. Federgruen and H. Groenevelt, "Characterization and optimization of achievable performance in general queueing systems," *Operations Research*, vol. 36, no. 5, pp. 733–741, 1988.

[44] M. Narasimhan and J. A. Bilmes, "Local search for balanced submodular clusterings." in *Proceedings of the 20th international joint conference on Artificial Intelligence*, ser. IJCAI, 2007, pp. 981–986.

[45] A. Krause and C. E. Guestrin, "Near-optimal nonmyopic value of information in graphical models," *arXiv preprint arXiv:1207.1394*, 2012.

[46] R. K. Iyer, "Submodular optimization and machine learning: Theoretical results, unifying and scalable algorithms, and applications," Ph.D. dissertation, University of Washington, 2015.

[47] F. Bach, "Learning with submodular functions: A convex optimization perspective," *Foundations and Trends in Machine Learning*, vol. 62, no. 2-3, pp. 145–373, 2013.

[48] M. Grötschel, L. Lovász, and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization," *Combinatorica*, vol. 1, no. 2, pp. 169–197, 1981.

[49] A. Schrijver, "A combinatorial algorithm minimizing submodular functions in strongly polynomial time," *Journal of Combinatorial Theory, Series B*, vol. 80, no. 2, pp. 346–355, 2000.

[50] S. Iwata, L. Fleischer, and S. Fujishige, "A combinatorial strongly polynomial algorithm for minimizing submodular functions," *Journal of the ACM (JACM)*, vol. 48, no. 4, pp. 761–777, 2001.

[51] Y. T. Lee, A. Sidford, and S. C.-w. Wong, "A faster cutting plane method and its implications for combinatorial and convex optimization," in *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS, 2015, pp. 1049–1065.

[52] H. Jiang, "Minimizing convex functions with integral minimizers," in *Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA, 2021, pp. 976–985.

[53] N. J. A. Harvey, "Matchings, matroids and submodular functions," Ph.D. dissertation, Massachusetts Institute of Technology, 2008.

[54] A. Graur, T. Pollner, V. Ramaswamy, and S. M. Weinberg, "New query lower bounds for submodular function minimization," in *Proceedings of the 2020 Conference on Innovations in Theoretical Computer Science*, ser. ITCS, 2020, pp. 64:1–64:16.

[55] T. Lee, T. Li, M. Santha, and S. Zhang, “On the cut dimension of a graph,” in *Proceedings of the 36th Computational Complexity Conference*, ser. CCC, 2021, pp. 15:1–15:35.

[56] R. Diestel, *Graph Theory, 5th Edition*, ser. Graduate Texts in Mathematics. Springer, 2016, vol. 173.

[57] K. Kawarabayashi and M. Thorup, “Deterministic edge connectivity in near-linear time,” *Journal of the ACM (JACM)*, vol. 66, no. 1, pp. 4:1–4:50, 2019.

[58] M. Henzinger, S. Rao, and D. Wang, “Local flow partitioning for faster edge connectivity,” in *Proceedings of the 28th annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA, 2017, pp. 1919–1938.

[59] M. Ghaffari, K. Nowicki, and M. Thorup, “Faster algorithms for edge connectivity via random 2-out contractions,” in *Proceedings of the 31st annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA, 2020, pp. 1260–1279.

[60] T. Saranurak, “A Simple Deterministic Algorithm for Edge Connectivity,” in *Proceedings of the 2021 Symposium on Simplicity in Algorithms*, ser. SOSA, 2021.

[61] C. Beideman, K. Chandrasekaran, S. Mukhopadhyay, and D. Nanongkai, “Faster connectivity in low-rank hypergraphs via expander decomposition,” in *Proceedings of the 23rd International Conference on Integer Programming and Combinatorial Optimization*, ser. IPCO, 2022, pp. 70–83.

[62] C. Chekuri and K. Quanrud, “Isolating Cuts, (Bi-)Submodularity, and Faster Algorithms for Connectivity,” in *Proceedings of the 48th International Colloquium on Automata, Languages and Programming*, ser. ICALP, 2021, pp. 50:1–50:20.

[63] L. Chen, R. Kyng, Y. P. Liu, R. Peng, M. P. Gutenberg, and S. Sachdeva, “Maximum flow and minimum-cost flow in almost-linear time,” in *Proceedings of the 63rd Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS, 2022, pp. 612–623.

[64] S. Mukhopadhyay and D. Nanongkai, “A note on isolating cut lemma for submodular function minimization,” *arXiv preprint arXiv:2103.15724*, 2021.

[65] D. Karger and C. Stein, “A new approach to the minimum cut problem,” *Journal of the ACM (JACM)*, vol. 43, no. 4, pp. 601–640, July 1996.

[66] A. Gupta, E. Lee, and J. Li, “The Karger-Stein algorithm is optimal for k-cut,” in *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing*, ser. STOC, 2020, pp. 473–484.

[67] A. Gupta, D. Harris, E. Lee, and J. Li, “Optimal Bounds for the k-cut Problem,” *arXiv preprint arXiv:2005.08301*, 2020.

[68] Y. Kamidoi, N. Yoshida, and H. Nagamochi, "A Deterministic Algorithm for Finding All Minimum k-Way Cuts," *SIAM Journal on Computing*, vol. 36, no. 5, pp. 1329–1341, 2007.

[69] M. Thorup, "Fully-dynamic min-cut," *Combinatorica*, vol. 27, no. 1, pp. 91–127, 2007. [Online]. Available: https://doi.org/10.1007/s00493-007-0045-2

[70] M. Xiao, "An Improved Divide-and-Conquer Algorithm for Finding All Minimum k-Way Cuts," in *Proceedings of 19th International Symposium on Algorithms and Computation*, ser. ISAAC, 2008, pp. 208–219.

[71] C. Chekuri, K. Quanrud, and C. Xu, "LP relaxation and tree packing for minimum k-cut," *SIAM Journal on Discrete Mathematics (Preliminary version in SOSA 2019)*, vol. 34, no. 2, pp. 1334–1353, 2020.

[72] O. Goldschmidt and D. Hochbaum, "A Polynomial Algorithm for the k-cut Problem for Fixed k," *Mathematics of Operations Research (Preliminary version in FOCS 1988)*, vol. 19, no. 1, pp. 24–37, Feb 1994.

[73] K. Chandrasekaran and C. Chekuri, "Hypergraph k-cut for fixed k in deterministic polynomial time," in *Proceedings of the 61st Annual Symposium on Foundations of Computer Science*, ser. FOCS, 2020, pp. 810–821.

[74] C. Beideman, K. Chandrasekaran, and W. Wang, "Deterministic enumeration of all minimum k-cut-sets in hypergraphs for fixed k," in *Proceedings of the 33rd Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA, 2022.

[75] C. Beideman, K. Chandrasekaran, and W. Wang, "Counting and enumerating optimum cut sets for hypergraph k-partitioning problems for fixed k," in *Proceedings of the 49th International Colloquium on Automata, Languages and Programming*, ser. ICALP, 2022, pp. 16:1–16:18.

[76] M. Ehrgott, *Multicriteria optimization.* Springer Science & Business Media, 2005, vol. 491.

[77] C. H. Papadimitriou and M. Yannakakis, "On the approximability of trade-offs and optimal access of web sources," in *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS, 2000, pp. 86–92.

[78] A. Armon and U. Zwick, "Multicriteria global minimum cuts," *Algorithmica*, vol. 46, no. 1, pp. 15–26, 2006.

[79] H. Aissi, A. R. Mahjoub, and R. Ravi, "Randomized Contractions for Multiobjective Minimum Cuts," in *Proceedings of the 25th Annual European Symposium on Algorithms*, ser. ESA, 2017, pp. 6:1–6:13.

[80] P. J. Carstensen, "Complexity of some parametric integer and network programming problems," *Mathematical Programming*, vol. 26, no. 1, pp. 64–75, 1983.

[81] H. Aissi, A. R. Mahjoub, T. McCormick, and M. Queyranne, "Strongly polynomial bounds for multiobjective and parametric global minimum cuts in graphs and hypergraphs," *Mathematical Programming (Preliminary version in IPCO 2014)*, vol. 154, no. 1-2, pp. 3–28, 2015.

[82] D. Karger, "Enumerating parametric global minimum cuts by random interleaving," in *Proceedings of the 48th Annual ACM Symposium on Theory of Computing*, ser. STOC, 2016, pp. 542–555.

[83] C. Beideman, K. Chandrasekaran, and C. Xu, "Multicriteria cuts and size-constrained k-cuts in hypergraphs," *Mathematical Programming (Preliminary version in RANDOM 2020)*, 2021.

[84] M. Goemans, N. Harvey, S. Iwata, and V. Mirrokni, "Approximating submodular functions everywhere," in *Proceedings of the 20th annual ACM-SIAM Symposium on Discrete algorithms*, ser. SODA, 2009, pp. 535–544.

[85] M.-F. Balcan, N. Harvey, and S. Iwata, "Learning symmetric non-monotone submodular functions," in *NIPS Workshop on Discrete Optimization in Machine Learning*, ser. NIPS, 2012.

[86] Z. Svitkina and L. Fleischer, "Submodular Approximation: Sampling-based Algorithms and Lower Bounds," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1715–1737, 2011.

[87] A. Badanidiyuru, S. Dobzinski, H. Fu, R. Kleinberg, N. Nisan, and T. Roughgarden, "Sketching valuation functions," in *Proceedings of the 23rd annual ACM-SIAM Symposium on Discrete algorithms*, ser. SODA, 2012, pp. 1025–1035.

[88] N. Devanur, S. Dughmi, R. Schwartz, A. Sharma, and M. Singh, "On the Approximation of Submodular Functions," Preprint in arXiv: 1304.4948v1, 2013.

[89] C. Seshadri and J. Vondrák, "Is submodularity testable," *Algorithmica*, vol. 69, no. 1, pp. 1–25, 2014.

[90] V. Feldman and J. Vondrák, "Optimal Bounds on Approximation of Submodular and XOS Functions by Juntas," *SIAM Journal on Computing*, vol. 45, no. 3, pp. 1129–1170, 2016.

[91] M.-F. Balcan and N. J. Harvey, "Submodular functions: Learnability, structure, and optimization," *SIAM Journal on Computing*, vol. 47, no. 3, pp. 703–754, 2018.

[92] C. Beideman, K. Chandrasekaran, C. Chekuri, and C. Xu, "Approximate representation of symmetric submodular functions via hypergraph cut functions," in *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2022)*, 2022.

[93] H. Nagamochi and T. Ibaraki, "Computing edge-connectivity in multigraphs and capacitated graphs," *SIAM Journal on Discrete Mathematics*, vol. 5, no. 1, pp. 54–66, 1992.

[94] K. Kawarabayashi and M. Thorup, "Deterministic edge connectivity in near-linear time," in *Proceedings of the 47th Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC, 2015, pp. 665–674.

[95] A. Rubinstein, T. Schramm, and S. M. Weinberg, "Computing exact minimum cuts without knowing the graph," in *Proceedings of the 2018 Conference on Innovations in Theoretical Computer Science*, ser. ITCS, 2018, pp. 39:1–39:16.

[96] S. Forster, D. Nanongkai, L. Yang, T. Saranurak, and S. Yingchareonthawornchai, "Computing and testing small connectivity in near-linear time and queries via fast local cut algorithms," in *Proceedings of the 31st annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA, 2020, pp. 2046–2065.

[97] D. Nanongkai, T. Saranurak, and S. Yingchareonthawornchai, "Breaking quadratic time for small vertex connectivity and an approximation scheme," in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC, 2019, pp. 241–252.

[98] J. Li, D. Nanongkai, D. Panigrahi, T. Saranurak, and S. Yingchareonthawornchai, "Vertex connectivity in poly-logarithmic max-flows," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC, 2021, pp. 317–329.

[99] D. Nanongkai and T. Saranurak, "Dynamic spanning forest with worst-case update time: adaptive, las vegas, and $O(n^{1/2-\epsilon})$-time," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC, 2017, pp. 1122–1129.

[100] C. Wulff-Nilsen, "Fully-dynamic minimum spanning forest with improved worst-case update time," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC. ACM, 2017, pp. 1130–1143.

[101] T. Saranurak and D. Wang, "Expander decomposition and pruning: Faster, stronger, and simpler," in *Proceedings of the 30th annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA. SIAM, 2019, pp. 2616–2635.

[102] J. Chuzhoy, Y. Gao, J. Li, D. Nanongkai, R. Peng, and T. Saranurak, "A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond," in *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS. IEEE Computer Society, 2020.

[103] G. Goranci, H. Räcke, T. Saranurak, and Z. Tan, "The expander hierarchy and its applications to dynamic graph algorithms," in *Proceedings of the 32nd annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA, 2021, pp. 2212–2228.

[104] A. Bernstein, J. v. d. Brand, M. P. Gutenberg, D. Nanongkai, T. Saranurak, A. Sidford, and H. Sun, "Fully-dynamic graph sparsifiers against an adaptive adversary," *arXiv preprint arXiv:2004.08432*, 2020.

[105] A. Bernstein, M. P. Gutenberg, and T. Saranurak, "Deterministic decremental reachability, scc, and shortest paths via directed expanders and congestion balancing," in *FOCS*. IEEE Computer Society, 2020.

[106] M. Thorup, "Minimum k-way Cuts via Deterministic Greedy Tree Packing," in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, ser. STOC, 2008, pp. 159–166.

[107] T. Fukunaga, "Computing minimum multiway cuts in hypergraphs," *Discrete Optimization*, vol. 10, no. 4, pp. 371–382, 2013.

[108] D. W. Matula, "A linear time 2+epsilon approximation algorithm for edge connectivity," in *Proceedings of the 4th annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA, 1993, pp. 500–504.

[109] S. Guha, A. McGregor, and D. Tench, "Vertex and hyperedge connectivity in dynamic graph streams," in *SIGMOD/PODS*. ACM, 2015, pp. 241–247.

[110] Y. Dinitz, "Dinitz'algorithm: The original version and even's version," in *Theoretical computer science*. Springer, 2006, pp. 218–240.

[111] L. Babai, P. Frankl, and J. Simon, "Complexity classes in communication complexity theory," in *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS, 1986, pp. 337–347.

[112] E. Lawler, "Cutsets and Partitions of Hypergraphs," *Networks*, vol. 3, pp. 275–285, 1973.

[113] L. Zhao, H. Nagamochi, and T. Ibaraki, "Greedy splitting algorithms for approximating multiway partition problems," *Mathematical Programming*, vol. 102, no. 1, pp. 167–183, 2005.

[114] K. Okumoto, H. Nagamochi, and T. Ibaraki, "Divide-and-conquer algorithms for partitioning hypergraphs and submodular systems," *Algorithmica*, vol. 62, no. 3, pp. 787–806, 2012.

[115] M. Goemans and V. S. Ramakrishnan, "Minimizing submodular functions over families of sets," *Combinatorica*, vol. 15, pp. 499–513, 1995.

[116] A. Gupta, E. Lee, and J. Li, "The number of minimum k-cuts: improving the Karger-Stein bound," in *Proceedings of the 51st ACM Symposium on Theory of Computing*, ser. STOC, 2019, pp. 229–240.

[117] W. Cunningham, "Decomposition of submodular functions," *Combinatorica*, vol. 3, pp. 53–68, 1980.

[118] M. Henzinger and D. Williamson, "On the number of small cuts in a graph," *Information Processing Letters*, vol. 59, pp. 41–44, 1996.

[119] H. Nagamochi, K. Nishimura, and T. Ibaraki, "Computing all small cuts in an undirected network," *SIAM Journal on Discrete Mathematics*, vol. 10, no. 3, pp. 469–481, 1997.

[120] H. W. Hamacher, J.-C. Picard, and M. Queyranne, "Ranking the cuts and cut-sets of a network," *North Holland Mathematical Studies*, vol. 95, pp. 183–200, 1984.

[121] V. Vazirani and M. Yannakakis, "Suboptimal cuts: Their enumeration, weight and number (extended abstract)," in *Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, ser. ICALP, 1992, pp. 366–377.

[122] H. Nagamochi and T. Ibaraki, *Algorithmic Aspects of Graph Connectivity.* Cambridge University Press, Cambridge, 2008.

[123] W. Cunningham and J. Edmonds, "A combinatorial decomposition theory," *Canadian Journal of Mathematics*, vol. 32, pp. 734–765, 1980.

[124] S. Fujishige, "Canonical decompositions of symmetric submodular functions," *Discrete Applied Mathematics*, vol. 5, pp. 175–190, 1983.

[125] H. Saran and V. Vazirani, "Finding k Cuts within Twice the Optimal," *SIAM Journal on Computing*, vol. 24, no. 1, pp. 101–108, 1995.

[126] R. Ravi and A. Sinha, "Approximating k-cuts using network strength as a lagrangean relaxation," *European Journal of Operational Research*, vol. 186, no. 1, pp. 77–90, 2008.

[127] P. Manurangsi, "Inapproximability of Maximum Biclique Problems, Minimum k-Cut and Densest At-Least-k-Subgraph from the Small Set Expansion Hypothesis," *Algorithms*, vol. 11(1), p. 10, 2018.

[128] A. Gupta, E. Lee, and J. Li, "An FPT Algorithm Beating 2-Approximation for k-Cut," in *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA, 2018, pp. 2821–2837.

[129] A. Gupta, E. Lee, and J. Li, "Faster exact and approximate algorithms for k-cut," in *Proceedings of the 59th IEEE annual Symposium on Foundations of Computer Science*, ser. FOCS, 2018, pp. 113–123.

[130] D. Lokshtanov, S. Saurabh, and V. Surianarayanan, "A Parameterized Approximation Scheme for Min k-Cut," in *Proceedings of the 61st IEEE annual Symposium on Foundations of Computer Science*, ser. FOCS, 2020, pp. 798–809.

[131] M. Xiao, "Finding minimum 3-way cuts in hypergraphs," *Information Processing Letters (Preliminary version in TAMC 2008)*, vol. 110, no. 14, pp. 554–558, 2010.

[132] E. Dahlhaus, D. Johnson, C. Papadimitriou, P. Seymour, and M. Yannakakis, "The complexity of multiterminal cuts," *SIAM Journal on Computing*, vol. 23, no. 4, pp. 864–894, 1994.

[133] M. Nägele, B. Sudakov, and R. Zenklusen, "Submodular minimization under congruency constraints," *Combinatorica*, vol. 39, pp. 1351–1386, 2019.

[134] K. Chandrasekaran and C. Chekuri, "Min-max partitioning of hypergraphs and symmetric submodular functions," in *Proceedings of the 32nd annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA, 2021, pp. 1026–1038.

[135] F. Guinez and M. Queyranne, "The size-constrained submodular k-partition problem," 2012, unpublished manuscript. Available at https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxmbGF2aW9ndWluZXpob21lcGFnZXxneDo0NDVlMThkMDg4ZWRlOGI1. See also https://smartech.gatech.edu/bitstream/handle/1853/43309/Queyranne.pdf.

[136] K. Mulmuley, "Lower bounds in a parallel model without bit operations," *SIAM Journal on Computing*, vol. 28, no. 4, pp. 1460–1509, 1999.

[137] M. Goemans and J. Soto, "Algorithms for symmetric submodular function minimization under hereditary constraints and generalizations," *SIAM Journal on Discrete Mathematics*, vol. 27, no. 2, pp. 1123–1145, 2013.

[138] K. Chandrasekaran, C. Xu, and X. Yu, "Hypergraph k-cut in randomized polynomial time," in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA, 2018, pp. 1426–1438.

[139] P. McMullen, "The maximum number of facets of a convex polytope," *Mathematika*, vol. 17, no. 2, pp. 179–184, 1970.

[140] W. H. Cunningham, "Minimum cuts, modular functions, and matroid polyhedra," *Networks*, vol. 15, no. 2, pp. 205–215, 1985.

[141] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency.* Springer Science & Business Media, 2003.

[142] Y. Chen, S. Khanna, and A. Nagda, "Near-linear size hypergraph cut sparsifiers," in *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS, 2020, pp. 61–72.

[143] S. Fujishige, *Submodular functions and optimization.* Elsevier, 2005.

[144] T. Helgason, "Aspects of the theory of hypermatroids," in *Hypergraph Seminar.* Springer, 1974, pp. 191–213.