Multicriteria Cuts and Size-Constrained k-Cuts in Hypergraphs

Calvin Beideman* Karthekeyan Chandrasekaran* Chao Xu†

Abstract

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 [1]. 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 $b \in \mathbb{R}_+^t$ 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 [10]. Our technique also shows that the number of optimal solutions is polynomial.

All of our results build on the random contraction approach of Karger [11]. 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.

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 $c_1, \ldots, c_t : E \to \mathbb{Z}_+$ be t non-negative hyperedge-cost functions, where t is a constant. The cost of a subset F of hyperedges under criterion $i \in [t]$ is $c_i(F) := \sum_{e\in F} c_i(e)$. For a positive integer k, a subset of hyperedges that crosses a k-partition $(U_1, \ldots, U_k)$ of the vertex set is said to be a k-cut. 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. A cut F is a *parametric min-cut* if there exist positive 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 $e \in E$. A cut F *dominates* another cut F' if $c_i(F) \le c_i(F')$ for every $i \in [t]$ and there exists $i \in [t]$ such that

*University of Illinois, Urbana-Champaign. Email: {`calvinb2, karthe`}`@illinois.edu`. Supported in part by NSF CCF-1907937 and CCF-1814613.

†The Voleon Group. Email: `the.chao.xu@gmail.com`.

$c_i(F) < c_i(F')$. A cut F is *pareto-optimal* if it is not dominated by any other cut. For a budget-vector $b \in \mathbb{R}_+^{t-1}$, 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 $b \in \mathbb{R}_+^{t-1}$ for which F is a b-multiobjective min-cut. These three notions satisfy the following relationship with the containment being possibly strict (see Appendix A.1 for a proof):

$$\text{Parametric min-cuts} \subseteq \text{Pareto-optimal cuts} \subseteq \text{Multiobjective min-cuts}. \tag{1}$$

There is also a natural notion of min-cuts under node-weighted budget constraints. Let $w_1, \ldots, w_t : V \to \mathbb{R}_+$ be vertex-weight functions and $c : E \to \mathbb{R}_+$ be a hyperedge-cost function. For a budget-vector $b \in \mathbb{R}_+^t$, 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 work, 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 [6] shows that the number of min-cuts in an n-vertex graph is $O(n^2)$ (also see Karger [11]). The same upper bound was shown to hold for constant-rank hypergraphs by Kogan and Krauthgamer [13] and for arbitrary-rank hypergraphs by Chekuri and Xu [5] and by Ghaffari, Karger, and Panigrahi [7] via completely different techniques. For $t = 2$ criteria in graphs, Mulmuley [14] 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 [1] 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 [12] 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 [3] showed that all pareto-optimal cuts in graphs can be enumerated in pseudo-polynomial time. For $t = 2$ criteria in constant-rank hypergraphs, Aissi et al [1] 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. Aissi et al 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 (1), answering our first question affirmatively would also answer their open question. On a related note, Aissi, Mahjoub, and Ravi [2] 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 [3] gave a polynomial-time algorithm to find a minimum valued cut with at most b vertices in the smaller side. Goemans and Soto [8] 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 [2] 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 $b \in \mathbb{R}_+^t$ 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 [9]. So, our focus is on constant k and constant sizes. Guinez and Queyranne [10] raised size-constrained min-k-cut with unit vertex-weights as a sub-problem 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.[1] 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 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 [4] via a non-uniform random contraction algorithm.

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.

[1]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.

Our first result is a strongly polynomial upper bound on the number of multiobjective min-cuts in constant-rank hypergraphs.

Theorem 1.1. *The number of multiobjective min-cuts in an r-rank, n-vertex hypergraph G 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 1.1 and Proposition 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 [1]. We also design randomized polynomial time algorithms to enumerate all multiobjective min-cuts and all pareto-optimal cuts in constant-rank hypergraphs (see Section 2.3). Independent of our work, Rico Zenklusen has also shown Theorem 1.1. We learned after 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.

Given the upper bound in Theorem 1.1, a discussion on the lower bound is in order. We recall that Karger [12] 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 (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 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 1.2.
1. *The number of node-budgeted multiobjective min-cuts in an r-rank, n-vertex hypergraph with t vertex-weight functions is $O(r2^rn^{t+2})$.*
2. *For a fixed budget-vector $b \in \mathbb{R}_+^t$, the number of node-budgeted b-multiobjective min-cuts in an n-vertex hypergraph with t vertex-weight functions is $O(n^2)$.*

We draw the reader's attention to the distinction between the two parts in Theorem 1.2. 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 $b \in \mathbb{R}_+^t$ 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 1.3. *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 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 1.3 resolves an open problem posed by Guinez and Queyranne [10]. A structural consequence of Theorem 1.3 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 1 for a comparison of known results and our contributions.

Problem	**Graphs**	**r-rank Hypergraphs**	**Hypergraphs**
# of Parametric Min-Cuts	$O(n^{t+1})$ [12] $\Omega(n^{t/2})$ [12]	$O(2^r n^{t+1})$ [12]	OPEN
# of Pareto-Optimal Cuts	$\tilde{O}(n^5)$ for $t = 2$ [1] $O(n^{3t-1})$ [Thm 1.1] $\Omega(n^t)$ [Thm 2.4]	$O(r2^{rt}n^{3t-1})$ [Thm 1.1]	OPEN
# of b-Multiobjective Min-Cuts	$O(n^{2t})$ [2] $\Omega(n^t)$ [Thm 2.4]	$O(r2^{rt}n^{2t})$ [Thm 2.1]	OPEN
# of Multiobjective Min-Cuts	$O(n^{3t-1})$ [Thm 1.1] $\Omega(n^t)$ [Thm 2.4]	$O(r2^{rt}n^{3t-1})$ [Thm 1.1]	OPEN
# of Node-Budgeted b-Multiobjective Min-Cuts	$O(n^2)$ [2]	$O(n^2)$ [Thm 1.2]	$O(n^2)$ [Thm 1.2]
# of Node-Budgeted Multiobjective Min-Cuts	$O(n^{t+2})$ [Thm 1.2]	$O(r2^r n^{t+2})$ [Thm 1.2]	OPEN
Node-Weighted s-Size-Constrained k-cut (const. k and const. $s \in \mathbb{Z}^k$)	Poly-time [10]	Poly-time [10]	Poly-time [Thm 1.3]

Table 1: Text in gray refers to known results while text in black illustrates results from this work. 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.

1.2 Technical Overview

As mentioned earlier, all our results build on the random contraction technique introduced by Karger [11] 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 [13] 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 n_1 say), then a certain number of contractions based on π_2 (until the number of vertices is n_2 say), and so on. So, the order of contractions becomes irrelevant and only the number 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 1.1, 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 [2] designed a random-contraction based algorithm to solve the b-multiobjective min-cut problem in graphs. Their algorithm proceeds as follows: Let $U_0 := \emptyset$ and 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 random contraction based algorithm that we know for arbitrary-rank hypergraphs crucially requires 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 1.2. 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 1.3. 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 [4] 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 2, we bound the number of multiobjective min-cuts (and prove Theorem 1.1), 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 3, we address node-budgeted multiobjective min-cuts and prove Theorem 1.2. In Section 4, we give an algorithm to solve size-constrained min-k-cut, thereby proving Theorem 1.3.

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. 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. We present its proof in the appendix.

Lemma 1.4. *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 natural numbers. Then, the optimum value of the linear program (LP_1) defined below is* $\min_{2 \leq j \leq r}(1 - \frac{j}{\gamma - r + j})f(n - j + 1)$.

$$\begin{aligned}
\underset{x_2,\ldots,x_r,y_2,\ldots,y_r}{\textit{minimize}} \quad & \sum_{j=2}^{r}(x_j - y_j)f(n - j + 1) && (LP_1)\\
\textit{subject to} \quad & 0 \leq y_j \leq x_j \ \forall j \in \{2, \ldots, r\} && (2)\\
& \sum_{j=2}^{r} x_j = 1 && (3)\\
& \gamma \sum_{j=2}^{r} y_j \leq \sum_{j=2}^{r} j \cdot x_j && (4)
\end{aligned}$$

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 1.1.

Let $G = (V, E)$ be an r-rank hypergraph and let $c_1, \ldots, c_t : E \to \mathbb{R}_+$ be cost functions on the hyperedges of G. For a subset F of hyperedges, we will use $c_i(F)$ to denote $\sum_{e \in F} c_i(e)$. For a subset U of vertices, we will use $\overline{U}$ to denote $V \setminus U$ and $\delta(U)$ to denote the set of hyperedges that intersect both U and $\overline{U}$. For a vertex v, we will use $\delta(v)$ to denote $\delta(\{v\})$. A subset F of hyperedges is a cut if there exists a partition $(U, \overline{U})$ such that $F = \delta(U)$. We refer the reader to Section 1 for the definitions of b-multiobjective min-cuts, multiobjective min-cuts, and pareto-optimal cuts.

We begin with a randomized algorithm for the b-multiobjective min-cut problem in Section 2.1. We take an alternative viewpoint of this randomized algorithm in Section 2.2 to prove Theorem 1.1. Since all pareto-optimal cuts are multiobjective min-cuts, Theorem 1.1 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 2.3, we give randomized polynomial-time algorithms to enumerate all pareto-optimal cuts and all multiobjective min-cuts. In Section 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.

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. We use Algorithm 1. We summarize its correctness and run-time guarantees in Theorem 2.1.

b-Multiobjective Min-Cut

Given: A hypergraph $G = (V, E)$ with hyperedge-cost functions $c_1, \ldots, c_t : E \to \mathbb{R}_+$ and a budget-vector $b \in \mathbb{R}_+^{t-1}$.

Goal: A b-multiobjective min-cut.

```
$b$-Multiobjective-Min-Cut($G, r, t, c, b$):
  Input:  An $r$-rank hypergraph $G = (V, E)$, hyperedge-cost functions
          $c_1, \ldots, c_t : E \to \mathbb{R}_+$ and a budget-vector $b \in \mathbb{R}_+^{t-1}$.
  If $|V| \le rt$:
      $X \leftarrow$ 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 1: b-Multiobjective Min-Cut

Theorem 2.1. *Let $G = (V, E)$ be an r-rank n-vertex hypergraph with hyperedge-cost functions $c_1, \ldots, c_t : E \to \mathbb{R}_+$ and let $b \in \mathbb{R}_+^{t-1}$ be a budget-vector. Let F be an arbitrary b-multiobjective min-cut. Then, Algorithm 1 outputs F with probability at least Q_n, where*

$$Q_n := \begin{cases} \frac{1}{2^{rt}} & \text{if } n \le rt, \text{and} \\ \frac{2t+1}{2^{rt}(rt+1)} \binom{n-t(r-2)}{2t}^{-1} & \text{if } n > rt. \end{cases}$$

Moreover, the algorithm can be implemented to run in polynomial time.

Proof. We note that 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 now bound the correctness probability by induction on n. For the base case, we consider $n \le rt$. 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 $\frac{1}{2^n} \ge \frac{1}{2^{rt}} = Q_n$.

Next, we prove the induction step. Suppose $n > rt$. We will need the following claim.

Claim 2.1. The algorithm outputs F with probability at least the optimum value of the following linear program.

$$\begin{aligned}
\underset{x_2,\ldots,x_r,y_2,\ldots,y_r}{\text{minimize}} \quad & \sum_{j=2}^{r}(x_j - y_j)Q_{n-j+1} \\
\text{subject to} \quad & 0 \le y_j \le x_j \ \forall j \in \{2,\ldots,r\} \\
& \sum_{j=2}^{r} x_j = 1 \\
& |U_i| \sum_{j=2}^{r} y_j \le \sum_{j=2}^{r} j \cdot x_j
\end{aligned} \tag{LP_2}$$

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

$$\begin{aligned}
E_j &:= \{e \in E \colon |e| = j\}, \\
x_j &:= \Pr[e' \in E_j] = \frac{c_i(E_j)}{c_i(E)}, \text{ and} \\
y_j &:= \Pr[e' \in E_j \cap F] = \frac{c_i(E_j \cap F)}{c_i(E)}.
\end{aligned}$$

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

$$\Pr[\text{Algorithm returns the cut } F] \ge \sum_{j=2}^{r}(x_j - y_j)Q_{n-j+1}. \tag{5}$$

The values of x_j and y_j will depend on the structure of G. However we can deduce some relationships between them. Since $0 \le c_i(E_j \cap F) \le c_i(E_j)$ for every $j \in \{2,\ldots,r\}$, we know that

$$0 \le y_j \le x_j \text{ for every } j \in \{2,\ldots,r\}. \tag{6}$$

Moreover, x_j is the probability of picking a hyperedge of size j. Hence,

$$\sum_{j=2}^{r} x_j = 1. \tag{7}$$

Next, we show that for every $i \in [t]$ and every $v \in U_i$, we have

$$c_i(F) \le c_i(\delta(v)). \tag{8}$$

If $i < t$, then $c_i(F) \le b_i < c_i(\delta(v))$ 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 again $c_i(F) \le c_i(\delta(v))$ for every $v \in U_i$.

From inequality (8), we conclude that

$$c_i(F) \le \frac{\sum_{v\in U_i} c_i(\delta(v))}{|U_i|} \le \frac{\sum_{v\in V} c_i(\delta(v))}{|U_i|} = \frac{\sum_{e\in E} |e| c_i(e)}{|U_i|} = \frac{\sum_{j=2}^{r} j\cdot c_i(E_j)}{|U_i|}.$$

Therefore

$$\sum_{j=2}^{r} y_j = \Pr[e' \in F] = \frac{c_i(F)}{c_i(E)} \le \frac{1}{|U_i|}\sum_{j=2}^{r} j\cdot x_j.$$

Thus, we have that

$$|U_i|\sum_{j=2}^{r} y_j \le \sum_{j=2}^{r} j\cdot x_j. \tag{9}$$

The minimum value of our lower bound in equation (5) over all choices of x_j and y_j that satisfy inequalities (6), (7), and (9) is a lower bound on the probability that the algorithm outputs F. ◻

Let U_i be a the largest among the sets $U_1,\dots,U_t$ that the algorithm generates when executed on input (G,r,t,c,b). Claim 2.1 tells us that the algorithm outputs F with probability at least the optimum value of the linear program (LP_2) from the claim.

The linear program (LP_2) is exactly the linear program (LP_1) from Lemma 1.4 with $\gamma = |U_i|$ and $f(n) := Q_n$. To apply Lemma 1.4, we just need to show that $n \ge |U_i| \ge r+1$. 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 $|U_i| \le |V| = n$. We also know from the the construction of the sets $U_1,\dots,U_t$ 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 $|U_i| \ge \frac{n}{t}$. Since $n > rt$, this means $|U_i| \ge \frac{rt+1}{t} > r$. Thus $|U_i| > r$, and since r and $|U_i|$ are integers, we conclude that $|U_i| \ge r+1$. Therefore, we can apply Lemma 1.4 with $\gamma = |U_i|$ to conclude that

$$\Pr[\text{Algorithm returns the cut } F] \ge \min_{2\le j\le r}\left(1 - \frac{j}{|U_i|-r+j}\right) Q_{n-j+1}.$$

The following claim completes the proof of the theorem. ◻

Claim 2.2. For every $j \in \{2,\dots,r\}$, we have

$$\left(1 - \frac{j}{|U_i|-r+j}\right) Q_{n-j+1} \ge Q_n$$

Proof. Let $j \in \{2,\dots,r\}$. The given inequality is equivalent to $\frac{Q_{n-j+1}}{Q_n} \ge \frac{|U_i|-r+j}{|U_i|-r}$. Since U_i is the largest among $U_1,\dots,U_t$ which together partition V, we have $|U_i| \ge \frac{n}{t}$. Consequently, $\frac{|U_i|-r+j}{|U_i|-r} = 1 + \frac{j}{|U_i|-r} \le 1 + \frac{jt}{n-rt}$. Therefore, it suffices to prove that $\frac{Q_{n-j+1}}{Q_n} \ge 1 + \frac{jt}{n-tr}$. We case on the value of $n-j+1$.

Case 1: Suppose that $n-j+1 > rt$. Then, we have

$$\frac{Q_{n-j+1}}{Q_n} = \frac{\binom{n-t(r-2)}{2t}}{\binom{n-j+1-t(r-2)}{2t}} = \prod_{\ell=0}^{2t-1} \frac{n-t(r-2)-\ell}{n-j+1-t(r-2)-\ell}. \tag{10}$$

We consider two sub-cases based on the value of j.

Case 1.a: Suppose that $j > 2t$. Then, we observe that

$$\begin{aligned}\prod_{\ell=0}^{2t-1} \frac{n-t(r-2)-\ell}{n-j+1-t(r-2)-\ell} &\geq \left(\frac{n-t(r-2)}{n-j+1-t(r-2)}\right)^{2t} \\ &= \left(1+\frac{j-1}{n-j+1-t(r-2)}\right)^{2t} \\ &\geq 1+\frac{2t(j-1)}{n-j+1-t(r-2)} \\ &\geq 1+\frac{jt+(j-2)t}{n-rt} \\ &\geq 1+\frac{jt}{n-rt}.\end{aligned}$$

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 the right hand side of equation (10) to obtain that

$$\begin{aligned}\prod_{\ell=0}^{2t-1} \frac{n-t(r-2)-\ell}{n-j+1-t(r-2)-\ell} &= \prod_{\ell=0}^{j-2} \frac{n-t(r-2)-\ell}{n-tr-\ell} \\ &\geq \left(\frac{n-t(r-2)}{n-tr}\right)^{j-1} \\ &= \left(1+\frac{2t}{n-rt}\right)^{j-1} \\ &\geq 1+\frac{2t(j-1)}{n-rt} \\ &\geq 1+\frac{jt}{n-rt}.\end{aligned}$$

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

$$\begin{aligned}\frac{Q_{n-j+1}}{Q_n} &= \frac{(rt+1)\binom{n-t(r-2)}{2t}}{2t+1} \\ &\geq \frac{(rt+1)\binom{rt+1-t(r-2)}{2t}}{2t+1} \\ &= rt+1 \\ &\geq 1+jt \\ &\geq 1+\frac{jt}{n-rt}.\end{aligned}$$

Thus, our desired inequality holds in all cases. ☐

2.2 Finding Multiobjective Min-Cuts

In this section, we present Algorithm 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 1, all of the randomness in Algorithm 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 2.2 summarizes the guarantees of Algorithm 2.

```
MULTIOBJECTIVE-MIN-CUT($G, r, t, c_1, \dots, c_t$):
  Input:  An $r$-rank hypergraph $G = (V, E)$ and
          hyperedge-cost functions $c_1, \dots, c_t : E \to \mathbb{R}_+$.

  If $|V| \le rt$:
      Pick a random subset $X$ of $V$ and return $\delta(X)$
  For $i = 1, \dots, t$:
      $E_i \leftarrow \{e \in E \colon c_i(e) > 0\}$
      $\pi_i \leftarrow$ 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 $n_1, \dots n_t$ with $n \ge n_1 \ge n_2 \ge \cdots \ge n_{t-1} \ge n_t = rt$:
      $G' \leftarrow G$
      For $i = 1, \dots, t$:
          While $|V(G')| > n_i$:
              $e \leftarrow$ 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 2: MULTIOBJECTIVE MIN-CUT

Theorem 2.2. *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 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 $n_1, \dots, n_{t-1} \in [n]$ (or just one cut if $|V| \le rt$). Hence, it outputs at most n^{t-1} cuts.

We now argue that the algorithm retains the same success probability as Algorithm 1, for any fixed budget-vector b.

Suppose $n \le rt$. Then both Algorithm 2 and Algorithm 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 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 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 1 so that it begins by selecting permutations $\pi_1, \dots, \pi_t$ of $E_1, \dots, E_t$ (where $E_i = \{e \in E \colon c_i(e) > 0\}$) as in Algorithm 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 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 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 2 is running exactly the version of Algorithm 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 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 1. By Theorem 2.1, this probability is $\frac{\Omega(n^{-2t})}{r2^{rt}}$, as desired. $\square$

We derive Theorem 1.1 from Theorem 2.2 now.

Theorem 1.1. *The number of multiobjective min-cuts in an r-rank, n-vertex hypergraph G with t hyperedge-cost functions is $O(r2^{rt}n^{3t-1})$.*

Proof. Let x be the number of multiobjective min-cuts in G. By Theorem 2.2, the expected number of multiobjective min-cuts output by our algorithm MULTIOBJECTIVE MIN-CUT (i.e., Algorithm 2) is $(x/r2^{rt})\Omega(n^{-2t})$. Theorem 2.2 also tells us that the algorithm outputs at most n^{t-1} cuts. Therefore, $x = r2^{rt} \cdot O(n^{3t-1})$. $\square$

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 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 $c_i(F') \leq c_i(F)$ for every $1 \leq i \leq t-1$.

Next, we give a polynomial time algorithm to enumerate pareto-optimal cuts. By containment relation (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 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($G, r, t, c_1, \dots, c_t, F$):
    Input: An $r$-rank hypergraph $G = (V, E)$, cost functions $c_1, \dots, c_t : E \to \mathbb{R}_+$,
           and a cut $F$ in $G$

    For $i = 1, \dots, t$:
        $\vec{c} \leftarrow (c_1, \dots c_{i-1}, c_{i+1}, \dots, c_t, c_i)$
        $\vec{b} \leftarrow (c_1(F), \dots, c_{i-1}(F), c_{i+1}(F), \dots, c_t(F))$
        For $j = 1, \dots, r2^{rt}O(n^{2t}\log(n))$:
            $F' \leftarrow$ $b$-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 3: Verify pareto-optimality of a given cut

Theorem 2.3. *Given an r-rank, n-vertex hypergraph G with t hyperedge-cost functions and a cut F in G, if F is a pareto-optimal cut, then Algorithm 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 our algorithm for b-Multiobjective Min-Cut (i.e., Algorithm 1) is a polynomial-time algorithm. Algorithm 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 1) for this value of i returns a multiobjective min-cut, then the algorithm will return false. By Theorem 2.1, our algorithm for b-Multiobjective Min-Cut returns a b-multiobjective min-cut with probability $\frac{1}{r2^{rt}}\Omega(\frac{1}{n^{2t}})$. 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. □

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 [12]. We recall that every parametric min-cut is a pareto-optimal cut by the containment relation (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.

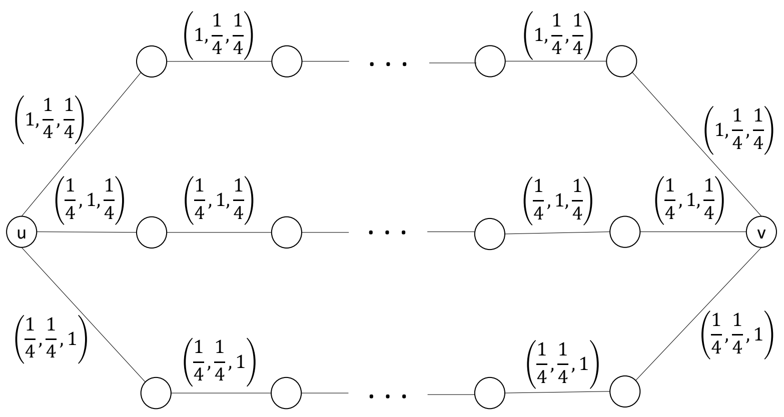

Figure 4: An illustration of our lower bound construction for $t = 3$.

Theorem 2.4. *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 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.

□

Remark 1. The lower bound from Theorem 2.4 is still significantly smaller than the $O(n^{3t-1})$ upper bound from Theorem 1.1. We believe that this gap comes from the slack in the analysis of our randomized algorithms.

Remark 2. We note that the construction in Theorem 2.4 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 2.4. Since distinct pareto-optimal cuts need not be b-multiobjective min-cuts for the same vector b, the bound in Theorem 2.4 does not immediately imply this bound either.

3 Node-Budgeted Multiobjective Min-Cuts

In this section, we give algorithms to find min-cuts that satisfy node-weighted budget constraints. Theorem 1.2 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 $w_1, \ldots, w_t : V \to \mathbb{R}_+$ 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 3.1. For a budget-vector $b \in \mathbb{R}^t_+$,

1. a vertex $v \in V$ is *feasible* if $w_i(v) \le b_i$ for all $i \in [t]$ and *infeasible* otherwise and
2. a set of vertices $S \subseteq V$ is *feasible* if $w_i(S) = \sum_{v \in S} w_i(v) \le b_i$ for all $i \in [t]$ and *infeasible* otherwise.

We recall the definition of node-budgeted multiobjective min-cuts.

Definition 3.2. For a budget-vector $b \in \mathbb{R}^t_+$, 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 $b \in \mathbb{R}^t_+$ 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 $w_1, \ldots, w_t \colon V \to \mathbb{R}_+$, a hyperedge-cost function $c \colon E \to \mathbb{R}_+$, and a budget-vector $b \in \mathbb{R}^t_+$.
Goal: A node-budgeted b-multiobjective min-cut.

3.1 Constant-Rank Hypergraphs

In this section, we design a polynomial-time algorithm to find node-budgeted b-multiobjective min-cuts in constant-rank hypergraphs and then prove the first part of Theorem 1.2. We use Algorithm 5 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 3.1. We will subsequently use Theorem 3.1 to prove the first part of Theorem 1.2.

Theorem 3.1. *Let $G = (V, E)$ be an r-rank n-vertex hypergraph with hyperedge-cost function $c : E \to \mathbb{R}_+$ and vertex-weight functions $w_1, \ldots, w_t : V \to \mathbb{R}_+$ and a budget-vector $b \in \mathbb{R}^t_+$. Let F be an arbitrary node-budgeted b-multiobjective min-cut in G. Then Algorithm 5 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

```
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 $|V| \leq r + 1$:
      $X \leftarrow$ 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 5: NODE-BUDGETED b-MULTIOBJECTIVE MIN-CUT in constant-rank hypergraphs

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 $n \leq r + 1$. 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 $\frac{1}{2^n} \geq \frac{1}{2^{r+1}} \geq Q_n$.

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 3.1. The algorithm outputs F with probability at least the optimum value of the following linear program.

$$\begin{aligned} \underset{x_2,\ldots,x_r,y_2,\ldots,y_r}{\text{minimize}} \quad & \sum_{j=2}^{r} (x_j - y_j) Q_{n-j+1} && (LP_3) \\ \text{subject to} \quad & 0 \leq y_j \leq x_j \ \forall j \in \{2, \ldots, r\} \\ & \sum_{j=2}^{r} x_j = 1 \\ & (n-1) \sum_{j=2}^{r} y_j \leq \sum_{j=2}^{r} j \cdot x_j \end{aligned}$$

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

$$
\begin{aligned}
E_j &:= \{e \in E \colon |e| = j\},\\
x_j &:= \Pr[e' \in E_j] = \frac{c(E_j)}{c(E)}, \text{ and}\\
y_j &:= \Pr[e' \in E_j \cap F] = \frac{c(E_j \cap F)}{c(E)}.
\end{aligned}
$$

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

$$
\Pr[\text{Algorithm returns the cut } F] \geq \sum_{j=2}^{r} (x_j - y_j) Q_{n-j+1} \tag{11}
$$

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

$$
0 \leq y_j \leq x_j \text{ for every } j \in \{2, \ldots, r\}. \tag{12}
$$

Moreover, x_j is the probability of picking a hyperedge of size j. Hence,

$$
\sum_{j=2}^{r} x_j = 1. \tag{13}
$$

We also know that $c(F) \leq c(\delta(v))$ for every feasible vertex v. Since we have assumed that G has at most 1 infeasible vertex, it has at least $n-1$ feasible ones, and thus,

$$
c(F) \leq \frac{\sum_{v : v \text{ is feasible}} c(\delta(v))}{|\{v \colon v \text{ is feasible}\}|} \leq \frac{\sum_{v \in V} c(\delta(v))}{n-1} = \frac{\sum_{e \in E} |e| c(e)}{n-1} = \frac{\sum_{j=2}^{r} j \cdot c(E_j)}{n-1}.
$$

Thus we have that,

$$
\sum_{j=2}^{r} y_j = \Pr[e' \in F] = \frac{c(F)}{c(E)} \leq \frac{1}{n-1} \sum_{j=2}^{r} j \cdot x_j. \tag{14}
$$

The minimum value of our lower bound in equation (11) over all choices of x_j and y_j that satisfy inequalities (12), (13), and (14) is a lower bound on the probability that the algorithm outputs F. □

Claim 3.1 tells us that the algorithm outputs F with probability at least the optimum value of the linear program (LP_3) from the claim. This linear program is exactly the linear program (LP_1) from Lemma 1.4 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 1.4 to conclude that

$$
\Pr[\text{Algorithm returns the cut } F] \geq \min_{j \in \{2, \ldots, r\}} \left(\left(1 - \frac{j}{n - 1 - r + j} \right) Q_{n-j+1} \right).
$$

It remains to show that $\min_{j \in \{2,\ldots,r\}}((1 - \frac{j}{n-1-r+j})Q_{n-j+1}) \geq Q_n$. Let $j \in \{2, \ldots, r\}$. 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

$$
\frac{Q_{n-j+1}}{Q_n} = \frac{\binom{n}{2}}{\binom{n-j+1}{2}} = \frac{n(n-1)}{(n-j+1)(n-j)} \geq \left(\frac{n}{n-j+1} \right)^2 = \left(1 + \frac{j-1}{n-j+1} \right)^2,
$$

and

$$\left(1+\frac{j-1}{n-j+1}\right)^2 \geq 1+\frac{2(j-1)}{n-j+1} \geq 1+\frac{j}{n-j+1} \geq 1+\frac{j}{n-1-r}.$$

The last two inequalities follow from the fact that $2 \leq j \leq r$. □

We now restate and prove the first part of Theorem 1.2. At a high level, our approach will be similar to the proof of Theorem 1.1. We will modify the algorithm for Node-Budgeted b-Multiobjective Min-Cut to obtain Algorithm 6 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 $w_i : V \to \mathbb{R}_+$ 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 $n' = 2, \ldots, n$:
      $G' \leftarrow G$
      Contract hyperedges from $G'$ in the order given by $\pi$ until $G'$ has at most $n'$ vertices
      For each $x_1, \ldots, x_t$ 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 6: Node-Budgeted Multiobjective Min-Cut in constant-rank hypergraphs

Theorem 3.2. *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 be an r-rank n-vertex hypergraph with vertex-weight functions $w_1, \ldots, w_t : V \to \mathbb{R}_+$. We will denote $w = (w_1, \ldots, w_t)$ and the hypergraph with the vertex-weight functions by the tuple (G, w) for conciseness. We first show that for any cut $F \subseteq E(G)$ which is a node-budgeted b-multiobjective min-cut in (G, w) for some budget-vector $b \in \mathbb{R}^t_+$, the cut F is among the cuts returned by Algorithm 6 with probability $\Omega(2^{-r}n^{-2})$.

We will view Algorithm 5 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 5 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 5 to use this method of contracting all infeasible vertices, and the success probability will still remain $\Omega(2^{-r}n^{-2})$.

Algorithm 6 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 $x_1, \ldots, x_t$ 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 6 will try the values of n' and $x_1, \ldots, x_t$ that the modified version of Algorithm 5 would use.

Therefore, by Theorem 3.1, 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 $O(r2^r n^{t+2})$.

□

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 1.2 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 [4] for the node-budgeted variant. In addition, our algorithm will use the non-uniform contraction algorithm for global min-cut by [4] as a subroutine. We reproduce their algorithm for completeness in Algorithm 7 and state its guarantee in Theorem 3.3.

```
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 7: Hypergraph Min-Cut

Theorem 3.3. *[4] Algorithm 7 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 to solve node-budgeted b-multiobjective min-cut in arbitrary-rank hypergraphs. We recall that a vertex v is feasible if $w_i(v) \leq b_i$ 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 3.1. Our algorithm chooses a hyperedge e to contract with probability proportional to

$$\alpha_e := \left(\frac{|U| - |e \cap U|}{|U|}\right) \cdot c(e) = \left(\frac{|U \setminus e|}{|U|}\right) \cdot c(e).$$

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 8. We summarize the correctness probability and the run-time of Algorithm 8 in Theorem 3.4.

Theorem 3.4. *Let G be an n-vertex hypergraph, for some $n \geq 2$ with vertex-weight functions $w_1, \ldots, w_t : V \to \mathbb{R}_+$, cost function $c\colon E \to \mathbb{R}_+$ and budget-vector $b \in \mathbb{R}^t_+$. Then Algorithm 8 outputs a fixed node-budgeted b-multiobjective min-cut in G with probability at least*

$$Q_n := \begin{cases} 1 & \text{if } n = 2, \\ \frac{1}{3}\binom{n-1}{2}^{-1} & \text{if } n \geq 3. \end{cases}$$

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.

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, $t \in \mathbb{Z}_+$, $w_1, \ldots, w_t : V(G) \to \mathbb{R}_+$, $c : E(G) \to \mathbb{R}_+$, and $b \in \mathbb{R}^t_+$. That is, $\mathcal{G}_n$ is the set of all valid inputs to Algorithm 8 where the hypergraph has n vertices. For an

```
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 $w_i : V \to \mathbb{R}_+$ for $i \in [t]$,
         a cost function $c : E \to \mathbb{R}_+$, and a budget-vector $b \in \mathbb{R}^t_+$
  $U \leftarrow \{v \in V : v \text{ is feasible}\}$
  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 8: NODE-BUDGETED b-MULTIOBJECTIVE MIN-CUT in arbitrary-rank hypergraphs.

input tuple T in the form just described, let $M(T)$ be the collection of b-multiobjective min-cuts for the input instance. Define

$$q_n := \min_{T \in \mathcal{G}_n} \min_{F \in M(T)} \Pr[\text{Algorithm returns } F \text{ on input } T].$$

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 3.3, the algorithm for HYPERGRAPH MIN-CUT (i.e., Algorithm 7) outputs F with probability at least $\binom{n}{2}^{-1}$. Consequently, Algorithm 8 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

$$\begin{aligned}
\Pr[\text{Algorithm returns } F \text{ on input } G] &= \sum_{e\in E\setminus F} \Pr[e'=e]\cdot \Pr[\text{Algorithm returns } F \text{ on input } G/e]\\
&\geq \sum_{e\in E\setminus F} \frac{\alpha_e}{\sum_{f\in E}\alpha_f}\cdot q_{n-|e|+1}\\
&\geq \frac{1}{\sum_{e\in E}\alpha_e}\sum_{e\in E\setminus F}\alpha_e Q_{n-|e|+1}.
\end{aligned}$$

Now, Claims 3.2 and 3.3 complete the proof of the theorem. □

Claim 3.2. 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

$$\begin{aligned}
\alpha_e Q_{n-|e|+1} &= \frac{|U\setminus e|c(e)}{|U|}\cdot\frac{1}{3}\binom{n-|e|}{2}^{-1}\\
&\geq \frac{|U|-|e|}{|U|}\cdot\frac{1}{3}\binom{n-|e|}{2}^{-1}\cdot c(e)\\
&\geq \frac{n-1-|e|}{n-1}\cdot\frac{2}{3(n-|e|)(n-|e|-1)}\cdot c(e)\\
&\geq \frac{2}{3(n-1)(n-|e|)}\cdot c(e)\\
&\geq \frac{2}{3(n-1)(n-2)}\cdot c(e)\\
&= c(e)Q_n.
\end{aligned}$$

□

Claim 3.3.

$$\frac{c(E\setminus F)}{\sum_{e\in E}\alpha_e} \geq 1.$$

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. $\square$

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 1.3. 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 partition $X = (X_1, \ldots, X_k)$ of V, we define $\delta(X)$ to be the set of hyperedges that intersect at least two parts of X. 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 4.1. 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 size-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.*

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-vector $s \in \mathbb{Z}_+^k$.

Goal: An 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

$$\alpha_e := \frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}}.$$

With these definitions, we solve s-size-constrained min-k-cut using Algorithm 9. We prove Theorem 1.3 using this algorithm.

Theorem 1.3. *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 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 $\sigma_{k-1} := \sum_{i=1}^{k-1} s_i$.

```
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 $s = (s_1, \ldots, s_k) \in \mathbb{Z}_+$

  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 9: s-SIZE-CONSTRAINED MIN-k-CUT

Proof. We consider Algorithm 9. 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

$$q_n := \min_{(G,w)\in\mathcal{G}_n} \min_{F\in M(G,w)} \Pr[\text{Algorithm returns } F \text{ on input } (G, w, k, s)], \text{ and}$$

$$Q_n := \begin{cases} \left(k^{\max\{2\sigma_{k-1},\sigma_k\}}\right)^{-1} & \text{if } n \leq \max\{2\sigma_{k-1}, \sigma_k\}, \textit{and} \\ \left(k^{\max\{2\sigma_{k-1},\sigma_k\}} n\binom{n}{2\sigma_{k-1}}\right)^{-1} & \text{if } n > \max\{2\sigma_{k-1}, \sigma_k\}. \end{cases}$$

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 $|Y_1| \leq |Y_2| \leq \cdots \leq |Y_k|$. 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 $n \leq \max\{2\sigma_{k-1}, \sigma_k\}$. 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, $\Pr[X = Y] \geq k^{-n} \geq k^{-max(2\sigma_{k-1},\sigma_k)} = Q_n$.

Now we will prove the inductive step. Assume that $n > \max\{2\sigma_{k-1}, \sigma_k\}$. By the inductive hypothesis, we have $q_{n'} \geq Q_{n'}$ for all $n' \in \{k, \ldots, n-1\}$. We will show that $q_n \geq Q_n$.

Suppose $|Y_k| \geq n - 2\sigma_{k-1}$. 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 $k^{-2\sigma_{k-1}} \geq Q_n$.

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 $w'(v) := \sum_{u \in e'} w(u)$ when 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

$$\begin{aligned}\Pr[R' = F \text{ on input } (G, k, s)] &= \sum_{e \in E \setminus F} \Pr[e' = e] \cdot \Pr[\text{Algorithm returns } F \text{ on input } (G/e, k, s)] \\ &\geq \sum_{e \in E \setminus F} \frac{\alpha_e}{\sum_{f \in E} \alpha_f} \cdot q_{n-|e|+1}.\end{aligned}$$

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,

$$\Pr[R' = F \text{ on input } (G, k, s)] \geq \frac{1}{\sum_{f \in E} \alpha_f} \sum_{e \in E \setminus F} \alpha_e \cdot Q_{n-|e|+1}.$$

We need the following two claims. We defer their proofs to complete the proof of the theorem.

Claim 4.1. For every hyperedge $e \in E \setminus F$, we have $\alpha_e Q_{n-|e|+1} \geq \frac{nQ_n}{n-1}$.

Claim 4.2. $\frac{|E \setminus F|}{\sum_{f \in E} \alpha_f} \geq 1$.

By Claim 4.1, we have that

$$\begin{aligned}\Pr[R' = F \text{ on input } (G,k,s)] &= \frac{1}{\sum_{f\in E}\alpha_f}\sum_{e\in E\setminus F}\alpha_e \cdot Q_{n-|e|+1}\\ &\geq \frac{1}{\sum_{f\in E}\alpha_f}\sum_{e\in E\setminus F}\frac{nQ_n}{n-1}\\ &= \frac{|E\setminus F|}{\sum_{f\in E}\alpha_f}\cdot\frac{nQ_n}{n-1}.\end{aligned}$$

Thus, Claim 4.2 implies that

$$\Pr[R' = F \text{ on input } (G,k,s)] \geq \frac{nQ_n}{n-1}.$$

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[Algorithm returns F] $\geq Q_n$. □

Proof of Claim 4.1. Let $e \in E \setminus F$. We recall that F contains all large hyperedges and hence,

$$|e| < n - 2\sigma_{k-1}. \tag{15}$$

First, suppose that $n - |e| + 1 \leq \max\{2\sigma_{k-1}, \sigma_k\}$. Then, we have $Q_{n-|e|+1} = k^{-\max\{2\sigma_{k-1},\sigma_k\}}$. We consider two subcases.

Case 1: Suppose $n \geq 3\sigma_{k-1}$. Then

$$\frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}}Q_{n-|e|+1} \geq \frac{Q_{n-|e|+1}}{\binom{n}{\sigma_{k-1}}} \geq \left(k^{\max\{2\sigma_{k-1},\sigma_k\}}\binom{n}{2\sigma_{k-1}}\right)^{-1} = nQ_n \geq \frac{nQ_n}{n-|e|+1}.$$

The first inequality follows from inequality (15), and the last inequality follows from the fact that $\binom{n}{\sigma_{k-1}} \leq \binom{n}{2\sigma_{k-1}}$ for $n \geq 3\sigma_{k-1}$.

Case 2: Suppose $n < 3\sigma_{k-1}$. We recall that $n > 2\sigma_{k-1}$. By inequality (15), $n - |e| > 2\sigma_{k-1}$. Letting $n = 2\sigma_{k-1} + x$ for some $x \in \{1, \ldots \sigma_{k-1}\}$, we have

$$\frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}} \geq \frac{\binom{2\sigma_{k-1}}{\sigma_{k-1}}}{\binom{2\sigma_{k-1}+x}{\sigma_{k-1}}} = \frac{(2\sigma_{k-1})!(\sigma_{k-1}+x)!}{(2\sigma_{k-1}+x)!\sigma_{k-1}!} = \prod_{i=1}^{x}\frac{\sigma_{k-1}+i}{2\sigma_{k-1}+i} \geq \left(\frac{1}{2}\right)^x.$$

We also know that

$$\binom{n}{2\sigma_{k-1}}^{-1} = \binom{2\sigma_{k-1}+x}{2\sigma_{k-1}}^{-1} = \frac{(2\sigma_{k-1})!x!}{(2\sigma_{k-1}+x)!} = \prod_{i=1}^{x}\frac{i}{2\sigma_{k-1}+i} \leq \left(\frac{1}{2}\right)^x.$$

Therefore,

$$\frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}}Q_{n-|e|+1} \geq \left(\frac{1}{2}\right)^x Q_{n-|e|+1} \geq \frac{Q_{n-|e|+1}}{\binom{n}{2\sigma_{k-1}}} = \left(k^{\max\{2\sigma_{k-1},\sigma_k\}}\binom{n}{2\sigma_{k-1}}\right)^{-1} = nQ_n \geq \frac{nQ_n}{n-|e|+1}.$$

Next, suppose that $n - |e| + 1 > \max\{2\sigma_{k-1}, \sigma_k\}$. We have that

$$\begin{aligned}
\alpha_e Q_{n-|e|+1} &= \frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}} Q_{n-|e|+1} \\
&= \frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}} \cdot \frac{1}{\binom{n-|e|+1}{2\sigma_{k-1}}} \cdot \left((n-|e|+1)k^{\max\{2\sigma_{k-1},\sigma_k\}}\right)^{-1} \\
&= \frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}} \cdot \frac{1}{\binom{n-|e|+1}{2\sigma_{k-1}}} \cdot \frac{n}{n-|e|+1} \binom{n}{2\sigma_{k-1}} \cdot Q_n \\
&\geq \frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}} \cdot \frac{1}{\binom{n-|e|+1}{2\sigma_{k-1}}} \cdot \binom{n}{2\sigma_{k-1}} \cdot \frac{nQ_n}{n-1}.
\end{aligned}$$

The following proposition completes the proof. We defer its proof to the appendix.

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 of Claim 4.2. 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,

$$|F| \leq \mathbb{E}(|\delta(Z)|) = \sum_{e \in E} \Pr(e \in \delta(Z)).$$

Negating the inequality and adding $|E|$ to both sides gives

$$\begin{aligned}
|E \setminus F| &\geq \sum_{e \in E} (1 - \Pr(e \in \delta(Z))) \\
&= \sum_{e \in E} \Pr(e \notin \delta(Z)) \\
&= \sum_{e \in E} \sum_{i=1}^{k} \Pr(e \subseteq Z_i) \\
&\geq \sum_{e \in E} \frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}} \\
&= \sum_{e \in E} \alpha_e.
\end{aligned}$$

Thus, $|E \setminus F| / \sum_{f \in E} \alpha_f \geq 1$. □

Remark. 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 $i \in [t]$.

5 Conclusion and Open Problems

In this work, we illustrated the versatility of the random contraction technique by addressing multi-criteria 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 $b \in \mathbb{R}_+^{t-1}$ 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?

References

[1] H. Aissi, A. 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)*, 154(1-2):3–28, 2015.

[2] H. Aissi, A. Mahjoub, and R. Ravi. Randomized Contractions for Multiobjective Minimum Cuts. In *Proceedings of the 25th Annual European Symposium on Algorithms*, ESA, pages 6:1–6:13, 2017.

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

[4] K. Chandrasekaran, C. Xu, and X. Yu. Hypergraph k-cut in randomized polynomial time. *Mathematical Programming (Preliminary version in SODA 2018)*, Nov 2019.

[5] C. Chekuri and C. Xu. Minimum cuts and sparsification in hypergraphs. *SIAM Journal on Computing*, 47(6):2118–2156, 2018.

[6] E. Dinits, A. Karzanov, and M. Lomonosov. On the structure of a family of minimal weighted cuts in a graph. *Studies in Discrete Optimizatoin (in Russian)*, pages 290–306, 1976.

[7] 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*, SODA, pages 1101—-1114, 2017.

[8] M. Goemans and J. Soto. Algorithms for symmetric submodular function minimization under hereditary constraints and generalizations. *SIAM Journal on Discrete Mathematics*, 27(2):1123–1145, 2013.

[9] O. Goldschmidt and D. Hochbaum. A Polynomial Algorithm for the k-cut Problem for Fixed k. *Mathematics of Operations Research*, 19(1):24–37, Feb 1994.

[10] F. Guinez and M. Queyranne. The size-constrained submodular k-partition problem. 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`, 2012.

[11] 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*, SODA, page 21–30, 1993.

[12] D. Karger. Enumerating parametric global minimum cuts by random interleaving. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing*, STOC, pages 542–555, 2016.

[13] D. Kogan and R. Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS, pages 367–376, 2015.

[14] K. Mulmuley. Lower bounds in a parallel model without bit operations. *SIAM Journal on Computing*, 28(4):1460–1509, 1999.

A Appendix

A.1 Comparison of Parametric, Pareto-Optimal, and Multiobjective Cuts

We prove the containment relationship (1) here.

Proposition A.1. *The following containment relationship holds, possibly with the containment being strict:*

$$\textit{Parametric min-cuts} \subseteq \textit{Pareto-optimal cuts} \subseteq \textit{Multiobjective min-cuts}.$$

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 10 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 11 for an example).

□

A.2 Proof of Lemma 1.4

We restate and prove Lemma 1.4.

Lemma 1.4. *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 natural numbers. Then, the optimum value of the linear program (LP_1) defined below is* $\min_{2\leq j\leq r}(1 - \frac{j}{\gamma-r+j})f(n-j+1)$.

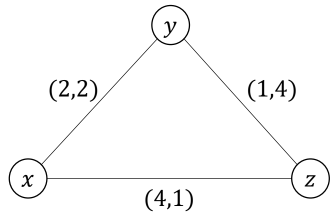

Figure 10: The cut $\delta(z)$ is a pareto-optimal cut but not a parametric min-cut.

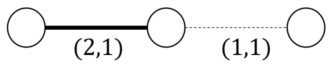

Figure 11: For $b = 2$, the bold edge is a b-multiobjective min-cut but it is not a pareto-optimal cut.

$$\begin{aligned}
&\underset{x_2,\ldots,x_r,y_2,\ldots,y_r}{\textit{minimize}} && \sum_{j=2}^{r}(x_j - y_j)f(n-j+1) && (LP_1)\\
&\textit{subject to} && 0 \le y_j \le x_j \ \forall j \in \{2,\ldots,r\} && (2)\\
& && \sum_{j=2}^{r} x_j = 1 && (3)\\
& && \gamma\sum_{j=2}^{r} y_j \le \sum_{j=2}^{r} j\cdot x_j && (4)
\end{aligned}$$

Proof. The linear program is feasible, since setting $x_2 = 1$ and the rest of the variables to zero gives a feasible solution. Let $j \in \{2,\ldots,r\}$. Since $y_j \ge 0$ and $x_j \le 1$, we have that $x_j - y_j \le 1$. Since $f(n-j+1) \ge 0$ for every $j \in [2,r]$, it follows that $(x_j - y_j)f(n-j+1) \le f(n-j+1)$. Therefore we have $\sum_{j=2}^{r}(x_j - y_j)f(n-j+1) \le \sum_{j=2}^{r} f(n-j+1)$. Therefore, the objective value of this linear program is bounded. Since the linear program is feasible and bounded, there exists an extreme point optimum solution to this LP. Since the LP has $2r-2$ variables and $2r$ equations, every extreme point optimum will have at least $2r-2$ tight constraints and at most 2 non-tight constraints.

We now show that constraint (4) is tight for every optimal solution (x,y). Let (x,y) be an optimal solution. Since $\gamma \ge r+1$, we have

$$\gamma\sum_{j=2}^{r} y_j \ge \sum_{j=2}^{r}(r+1)y_j > \sum_{j=2}^{r} j\cdot y_j.$$

This implies that we cannot have $y_j = x_j$ for all j, otherwise (x,y) would violate constraint (4). Hence, at least one of the $y_j \le x_j$ constraints must be slack. Let $j \in \{2,\ldots,r\}$ be such that $y_j < x_j$. If constraint (4) were 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, constraint (4) must be tight.

Let (x,y) be an extreme point optimal solution. Since we know that $\sum_{j=2}^{r} x_j = 1$ and $x_j \ge 0$ for every $j \in \{2,\ldots,r\}$, we have $\sum_{j=2}^{r} j\cdot x_j > 0$. Since constraint (4) 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 $0 \le y_{j_1}$ and $y_{j_2} \le x_{j_2}$ 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

$$\begin{aligned} \underset{y_j}{\text{minimize}} \quad & (1 - y_j) f(n - j + 1) \\ \text{subject to} \quad & 0 \le y_j \le 1 \\ & \gamma y_j = j. \end{aligned}$$

The only (and therefore optimal) solution to this LP is $y_j = \frac{j}{\gamma}$, which achieves an objective value of

$$\left(1 - \frac{j}{\gamma}\right) f(n - j + 1).$$

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

$$\begin{aligned} \underset{x_{j_1}}{\text{minimize}} \quad & (1 - x_{j_1}) f(n - j_2 + 1) \\ \text{subject to} \quad & 0 \le x_{j_1} \le 1 \\ & \gamma x_{j_1} = j_1 \cdot x_{j_1} + j_2 \cdot (1 - x_{j_1}). \end{aligned}$$

Solving the second constraint for x_{j_1} yields $x_{j_1} = \frac{j_2}{\gamma - j_1 + j_2}$, and therefore our optimal objective value is

$$\left(1 - \frac{j_2}{\gamma - j_1 + j_2}\right) f(n - j_2 + 1).$$

We conclude that the optimal objective value of the LP is equal to the minimum of the values from these two cases, that is,

$$\min\left\{ \min_{j \in \{2,\ldots,r\}} \left\{ \left(1 - \frac{j}{\gamma}\right) f(n - j + 1) \right\},\ \min_{j_1, j_2 \in \{2,\ldots,r\}} \left\{ \left(1 - \frac{j_2}{\gamma - j_1 + j_2}) f(n - j_2 + 1) \right) \right\} \right\}.$$

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

$$\min_{j_1, j_2 \in \{2,\ldots,r\}} \left\{ \left(1 - \frac{j_2}{\gamma - j_1 + j_2}\right) f(n - j_2 + 1) \right\} = \min_{j \in \{2,\ldots,r\}} \left\{ \left(1 - \frac{j}{\gamma - r + j}\right) f(n - j + 1) \right\}.$$

Thus, since $j \le r$, the optimal objective value of the LP is equal to

$$\min_{j \in \{2,\ldots,r\}} \left\{ \min\left\{ 1 - \frac{j}{\gamma}, 1 - \frac{j}{\gamma - r + j} \right\} f(n - j + 1) \right\} = \min_{j \in \{2,\ldots,r\}} \left\{ \left(1 - \frac{j}{\gamma - r + j}\right) f(n - j + 1) \right\}.$$

□

A.3 Proof of Proposition 4.1

We restate and prove Proposition 4.1.

Proposition 4.1. *For positive integers n, e, σ with $e \ge 2$ and $n - e + 1 > 2\sigma$, we have*

$$\frac{\binom{n-e}{\sigma}}{\binom{n}{\sigma}} \cdot \frac{1}{\binom{n-e+1}{2\sigma}} \ge \binom{n}{2\sigma}^{-1}.$$

Proof. We note that

$$\frac{\binom{n-e}{\sigma}}{\binom{n}{\sigma}} \cdot \frac{1}{\binom{n-e+1}{2\sigma}} = \frac{(n-e)!(n-\sigma)!}{n!(n-e-\sigma)!} \cdot \frac{(2\sigma)!(n-e-2\sigma+1)!}{(n-e+1)!}$$
$$= \left(\prod_{i=0}^{\sigma-1} \frac{n-e-i}{n-i}\right) \cdot \frac{(2\sigma)!}{\prod_{i=0}^{2\sigma-1}(n-e+1-i)}. \tag{16}$$

To lower bound this expression, we case on the value of e.

Case 1: Suppose $e > \sigma$. Then we can lower bound expression (16) by

$$\frac{1}{\prod_{i=0}^{\sigma-1}(n-i)} \cdot \frac{(2\sigma)!}{(n-e+1)\prod_{i=0}^{\sigma-2}(n-e-\sigma-i)} \geq \frac{(2\sigma)!}{\prod_{i=0}^{2\sigma-1}(n-i)} = \binom{n}{2\sigma}^{-1}.$$

Case 2: Suppose $e \leq \sigma$. We note that

$$\frac{(2\sigma)!}{\prod\limits_{i=0}^{2\sigma-1}(n-e+1-i)} = \frac{(2\sigma)!}{\prod\limits_{i=0}^{2\sigma-1}(n-i)} \cdot \prod_{i=0}^{e-2} \frac{n-i}{n-2\sigma-i} = \binom{n}{2\sigma}^{-1} \cdot \prod_{i=0}^{e-2} \frac{n-i}{n-2\sigma-i}.$$

Thus, expression (16) is equal to

$$\binom{n}{2\sigma}^{-1} \cdot \left(\prod_{i=0}^{\sigma-1} \frac{n-e-i}{n-i}\right)\left(\prod_{i=0}^{e-2} \frac{n-i}{n-2\sigma-i}\right).$$

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

$$\left(\prod_{i=0}^{\sigma-1} \frac{n-e-i}{n-i}\right)\left(\prod_{i=0}^{e-2} \frac{n-i}{n-2\sigma-i}\right) = \frac{\prod_{i=0}^{\sigma-1}(n-e-i)}{\prod_{i=e-1}^{\sigma-1}(n-i)} \cdot \frac{1}{\prod_{i=0}^{e-2}(n-2\sigma-i)}$$
$$= \frac{\prod_{i=\sigma}^{e+\sigma-1}(n-i)}{(n-e+1)\prod_{i=0}^{e-2}(n-2\sigma-i)}. \tag{17}$$

We claim that expression (17) is minimized when $e = 2$. To see this, we note that

$$\frac{\prod\limits_{i=\sigma}^{(e+1)+\sigma-1}(n-i)}{(n-(e+1)+1)\prod\limits_{i=0}^{(e+1)-2}(n-2\sigma-i)} = \frac{\prod\limits_{i=\sigma}^{e+\sigma-1}(n-i)}{(n-e+1)\prod\limits_{i=0}^{e-2}(n-2\sigma-i)} \cdot \frac{(n-e-\sigma)(n-e+1)}{(n-2\sigma-e+1)(n-e)}.$$

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 (17) increases when we increment e. Thus

$$\frac{\prod_{i=\sigma}^{e+\sigma-1}(n-i)}{(n-e+1)\prod_{i=0}^{e-2}(n-2\sigma-i)} \geq \frac{(n-\sigma)(n-\sigma-1)}{(n-1)(n-2\sigma)} = \frac{n^2-(2\sigma+1)n+(\sigma+1)\sigma}{n^2-(2\sigma+1)n+2\sigma} \geq 1.$$

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

$$\frac{\binom{n-e}{\sigma}}{\binom{n}{\sigma}} \cdot \frac{1}{\binom{n-e+1}{2\sigma}} \geq \binom{n}{2\sigma}^{-1}.$$

□