https://pubsonline.informs.org/journal/moor

MATHEMATICS OF OPERATIONS RESEARCH
Articles in Advance, pp. 1–23
ISSN 0364-765X (print), ISSN 1526-5471 (online)

Counting and Enumerating Optimum Cut Sets for Hypergraph *k*-Partitioning Problems for Fixed *k*

Calvin Beideman,[a] Karthekeyan Chandrasekaran,[a,*] Weihang Wang[a]

[a] University of Illinois Urbana-Champaign, Urbana, Illinois 61801
*Corresponding author
Contact: calvinb2@illinois.edu (CB); karthe@illinois.edu, https://orcid.org/0000-0002-3421-7238 (KC); weihang3@illinois.edu (WW)

Received: September 8, 2022
Revised: April 28, 2023
Accepted: October 30, 2023
Published Online in Articles in Advance: December 13, 2023

MSC2020 Subject Classifications: Primary: 68R10, 05C65

https://doi.org/10.1287/moor.2022.0259

Abstract. We consider the problem of enumerating optimal solutions for two hypergraph *k*-partitioning problems, namely, Hypergraph-*k*-Cut and Minmax-Hypergraph-*k*-Partition. The input in hypergraph *k*-partitioning problems is a hypergraph $G = (V, E)$ with positive hyperedge costs along with a fixed positive integer k. The goal is to find a partition of V into k nonempty parts $(V_1, V_2, \ldots, V_k)$—known as a *k*-partition—so as to minimize an objective of interest. (1) If the objective of interest is the maximum cut value of the parts, then the problem is known as Minmax-Hypergraph-*k*-Partition. A subset of hyperedges is a minmax-*k*-cut-set if it is the subset of hyperedges crossing an optimum *k*-partition for Minmax-Hypergraph-*k*-Partition. (2) If the objective of interest is the total cost of hyperedges crossing the *k*-partition, then the problem is known as Hypergraph-*k*-Cut. A subset of hyperedges is a min-*k*-cut-set if it is the subset of hyperedges crossing an optimum *k*-partition for Hypergraph-*k*-Cut. We give the first polynomial bound on the number of minmax-*k*-cut-sets and a polynomial-time algorithm to enumerate all of them in hypergraphs for every fixed k. Our technique is strong enough to also enable an $n^{O(k)}p$-time deterministic algorithm to enumerate all min-*k*-cut-sets in hypergraphs, thus improving on the previously known $n^{O(k^2)}p$-time deterministic algorithm, in which n is the number of vertices and p is the size of the hypergraph. The correctness analysis of our enumeration approach relies on a structural result that is a strong and unifying generalization of known structural results for Hypergraph-*k*-Cut and Minmax-Hypergraph-*k*-Partition. We believe that our structural result is likely to be of independent interest in the theory of hypergraphs (and graphs).

Funding: All authors were supported by NSF AF 1814613 and 1907937.

Keywords: hypergraphs • *k*-partition problems • algorithms • enumeration

1. Introduction

In hypergraph *k*-partitioning problems, the input consists of a hypergraph $G = (V, E)$ with positive hyperedge costs $c : E \to \mathbb{R}_+$ and a fixed positive integer k (e.g., $k = 2, 3, 4, \ldots$). The goal is to find a partition of the vertex set into k nonempty parts $V_1, V_2, \ldots, V_k$ so as to minimize an objective of interest. There are several natural objectives of interest in hypergraph *k*-partitioning problems. In this work, we focus on two particular objectives, Minmax-Hypergraph-*k*-Partition and Hypergraph-*k*-Cut:

1. In Minmax-Hypergraph-*k*-Partition, the objective is to minimize the maximum cut value of the parts of the *k*-partition—that is, minimize $\max_{i=1}^{k} c(\delta(V_i))$; here, $\delta(V_i)$ is the set of hyperedges intersecting both V_i and $V \setminus V_i$, and $c(\delta(V_i)) = \sum_{e \in \delta(V_i)} c(e)$ is the total cost of hyperedges in $\delta(V_i)$.
2. In Hypergraph-*k*-Cut,[1] the objective is to minimize the cost of hyperedges crossing the *k*-partition—that is, minimize $c(\delta(V_1, \ldots, V_k))$; here, $\delta(V_1, \ldots, V_k)$ is the set of hyperedges that intersect at least two sets in $\{V_1, \ldots, V_k\}$, and $c(\delta(V_1, \ldots, V_k)) = \sum_{e \in \delta(V_1, \ldots, V_k)} c(e)$ is the total cost of hyperedges in $\delta(V_1, \ldots, V_k)$.

If the input G is a graph, then we refer to these problems as Minmax-Graph-*k*-Partition and Graph-*k*-Cut, respectively. We note that the case of $k = 2$ corresponds to global minimum cut in both objectives. In this work, we focus on the problem of enumerating all optimum solutions to Minmax-Hypergraph-*k*-Partition and Hypergraph-*k*-Cut.

1.1. Motivations and Related Problems

We consider the problem of counting and enumerating optimum solutions for partitioning problems over hypergraphs for three reasons. First, hyperedges provide more powerful modeling capabilities than edges, and consequently, several problems in hypergraphs become nontrivial in comparison with graphs. Although hypergraphs and partitioning problems over hypergraphs (including Minmax-Hypergraph-*k*-Partition) are discussed as early as

1973 by Lawler [45], most of these problems still remain open. The powerful modeling capability of hyperedges has been useful in a variety of modern applications, which, in turn, has led to a resurgence in the study of hypergraphs with recent works focusing on min-cuts, cut-sparsifiers, spectral-sparsifiers, etc. (Bansal et al. [6], Beideman et al. [7], Chandrasekaran and Chekuri [13], Chandrasekaran et al. [16], Chekuri and Xu [18], Chen et al. [20], Fox et al. [24], Ghaffari et al. [25], Kapralov et al. [37], Kogan and Krauthgamer [44], Soma and Yoshida [58]). Our work adds to this rich and emerging theory of hypergraphs.

Second, hypergraph k-partitioning problems are special cases of submodular k-partitioning problems. In submodular k-partitioning problems, the input is a finite ground set V, a submodular function[2] $f : 2^V \to \mathbb{R}$ provided by an evaluation oracle,[3] and a positive integer k (e.g., $k = 2, 3, 4, \ldots$). The goal is to partition the ground set V into k nonempty parts $V_1, V_2, \ldots, V_k$ so as to minimize an objective of interest. Two natural objectives are of interest: (1) in Minmax-Submod-k-Partition, the objective is to minimize $\max_{i=1}^{k} f(V_i)$, and (2) in Minsum-Submod-k-Partition, the objective is to minimize $\sum_{i=1}^{k} f(V_i)$. If the given submodular function is symmetric,[4] then we denote the resulting problems as Minmax-SymSubmod-k-Partition and Minsum-SymSubmod-k-Partition, respectively. Because the hypergraph cut function is symmetric submodular, it follows that Minmax-Hypergraph-k-Partition is a special case of Minmax-Sym-Submod-k-Partition. Moreover, Hypergraph-k-Cut is a special case of Minsum-Submod-k-Partition (this reduction is slightly nontrivial with the submodular function in the reduction being asymmetric—e.g., see Okumoto et al. [52] for the reduction). Queyranne [54] claimed, in 1999, a polynomial-time algorithm for Minsum-SymSubmod-k-Partition for every fixed k; however, the claim was retracted subsequently (see Guinez and Queyranne [28]). The complexity status of submodular k-partitioning problems (for fixed $k \geq 4$) are open, so recent works focus on hypergraph k-partitioning problems as a stepping-stone toward submodular k-partitioning (Beideman et al. [7], Chandrasekaran and Chekuri [12, 13], Guinez and Queyranne [28], Okumoto et al. [52], Zhao [62], Zhao et al. [63]). Our work contributes to this stepping-stone by advancing the state of the art in hypergraph k-partitioning problems. We emphasize that the complexity status of two other variants of hypergraph k-partitioning problems that are also special cases of Minsum-Submod-k-Partition are still open: namely, Hypergraph-k-Partition in which the objective is to minimize the sum of the cut values of the k parts and Normalized-Coverage-k-Partition in which the objective is to minimize the sum of cost of hyperedges relative to the partition $\mathcal{P}$, where the cost of a hyperedge e relative to $\mathcal{P}$ is $c(e)(\ell - 1)$ with ℓ being the number of parts of $\mathcal{P}$ intersected by e (see Okumoto et al. [52], Zhao [62], Zhao et al. [63] for these variants).

Third, counting and enumeration of optimum solutions for graph k-partitioning problems are fundamental to graph theory and extremal combinatorics. They have found farther reaching applications than initially envisioned. We discuss some of the results and applications for $k = 2$ and $k > 2$ now. For $k = 2$ in connected graphs, it is well-known that the number of min-cuts and of α-approximate min-cuts are at most $\binom{n}{2}$ and $O(n^{2\alpha})$, respectively, and they can all be enumerated in polynomial time for constant α. These combinatorial results are the crucial ingredients of several algorithmic and representation results in graphs. On the algorithmic front, these results enable fast randomized construction of graph skeletons, which, in turn, play a crucial role in fast algorithms to solve graph min-cut (Karger [39]). On the representation front, counting results forms the backbone of cut sparsifiers, which, in turn, find applications in sketching and streaming (Ahn and Guha [2], Ahn et al. [3, 4], Kogan and Krauthgamer [44]). A polygon representation of the family of 6/5-approximate min-cuts in graphs was given by Benczur and Goemans in 1997 (see Benczur [9, 10], Benczur and Goemans [11]); this representation was used in the recent groundbreaking $(3/2 - \epsilon)$-approximation for traveling salesman problem (TSP) (Karlin et al. [41]). On the approximation front, in addition to the $(3/2 - \epsilon)$-approximation for metric TSP (Karlin et al. [41]), counting results also led to the recent 1.5-approximation for path TSP (Zenklusen [61]). For $k > 2$, we note that fast algorithms for Graph-k-Cut have been of interest because they help in generating cutting planes when solving TSP (Applegate et al. [5], Cook et al. [21]). A recent series of works aimed toward improving the bounds on the number of optimum solutions for Graph-k-Cut culminated in a drastic improvement in the runtime to solve Graph-k-Cut: namely, from $O(n^{2k})$ time to $O(n^k)$ time (Gupta et al. [31–33]. Given the status of counting and enumeration results for k-partitioning in graphs and their algorithmic and representation implications that were discovered subsequently, we believe that a similar understanding in hypergraphs could serve as an important ingredient in the algorithmic and representation theory of hypergraphs.

1.2. The Enumeration Problem

There is a fundamental structural distinction between hypergraphs and graphs that becomes apparent when enumerating optimum solutions to k-partitioning problems. In connected graphs, the number of optimum k-partitions for Graph-k-Cut and for Minmax-Graph-k-Partition are $n^{O(k)}$ and $n^{O(k^2)}$, respectively, and they can all be enumerated in polynomial time, in which n is the number of vertices in the input graph (Chandrasekaran and Wang [14],

Chekuri et al. [19], Gupta et al. [32, 33], Karger and Stein [40], Thorup [59]). In contrast, a connected hypergraph can have exponentially many optimum k-partitions for both MINMAX-HYPERGRAPH-k-PARTITION and HYPERGRAPH-k-CUT even for $k=2$; for example, consider the hypergraph with a single hyperedge containing all vertices, which we denote as the spanning hyperedge example. Hence, enumerating all optimum k-partitions for hypergraph k-partitioning problems in polynomial time is impossible. Instead, our goal in the enumeration problems is to enumerate k-cut sets corresponding to optimum k-partitions. We call a subset $F \subseteq E$ of hyperedges a k-cut set if there exists a k-partition $(V_1, \ldots, V_k)$ such that $F = \delta(V_1, \ldots, V_k)$; we call a 2-cut set a cut set. In the enumeration problems that we consider, the input consists of a hypergraph $G = (V, E)$ with positive hyperedge costs $c : E \to \mathbb{R}_+$ and a fixed positive integer k (e.g., $k = 2, 3, 4, \ldots$).

1. For an optimum k-partition $(V_1, \ldots, V_k)$ for MINMAX-HYPERGRAPH-k-PARTITION in (G, c), we denote $\delta(V_1, \ldots, V_k)$ as a MINMAX-k-CUT-SET. In ENUM-MINMAX-HYPERGRAPH-k-PARTITION, the goal is to enumerate all MINMAX-k-CUT-SETS.
2. For an optimum k-partition $(V_1, \ldots, V_k)$ for HYPERGRAPH-k-CUT in (G, c), we denote $\delta(V_1, \ldots, V_k)$ as a MIN-k-CUT-SET. In ENUM-HYPERGRAPH-k-CUT, the goal is to enumerate all MIN-k-CUT-SETS.

We observe that, in the spanning hyperedge example, although the number of optimum k-partitions for MINMAX-HYPERGRAPH-k-PARTITION (as well as HYPERGRAPH-k-CUT) is exponential, the number of MINMAX-k-CUT-SETS (as well as MIN-k-CUT-SETS) is only one.

1.3. Results

In contrast to graphs, whose representation size is the number of edges, the representation size of a hypergraph $G = (V, E)$ is $p := \sum_{e \in E} |e|$. Throughout, our algorithmic discussion focuses on the case of fixed k (e.g., $k = 2, 3, 4, \ldots$).

There are no prior results regarding ENUM-MINMAX-HYPERGRAPH-k-PARTITION in the literature. We recall the status of MINMAX-HYPERGRAPH-k-PARTITION. As mentioned earlier, MINMAX-HYPERGRAPH-k-PARTITION is discussed as early as 1973 by Lawler [45] with its complexity status being open until recently. We note that the objective here can be viewed as aiming to find a fair k-partition, that is, a k-partition in which no part pays too much in cut value. Motivated by this connection to fairness, Chandrasekaran and Chekuri [12] study the more general problem of MINMAX-SYMSUBMOD-k-PARTITION. They give the first (deterministic) polynomial-time algorithm to solve MINMAX-SYMSUBMOD-k-PARTITION and, as a consequence, obtain the first polynomial-time algorithm to solve MINMAX-HYPERGRAPH-k-PARTITION. Their algorithm does not show any bound on the number of MINMAX-k-CUT-SETS because it solves the more general problem of MINMAX-SYMSUBMOD-k-PARTITION for which the number of optimum k-partitions can indeed be exponential (recall the spanning hyperedge example). Focusing on hypergraphs raises the question of whether all k-cut sets corresponding to optimum solutions can be enumerated efficiently for every fixed k. We answer this question affirmatively by giving the first polynomial-time algorithm for ENUM-MINMAX-HYPERGRAPH-k-PARTITION.

Theorem 1. *There exists a deterministic algorithm to solve* ENUM-MINMAX-HYPERGRAPH-k-PARTITION *that runs in time* $O(kn^{4k^2-2k+1}p)$, *where* n *is the number of vertices and* p *is the size of the input hypergraph. Moreover, the number of* MINMAX-K-CUT-SETS *in a* n*-vertex hypergraph is* $O(n^{4k^2-2k})$.

We emphasize that our result shows the first polynomial bound on the number of MINMAX-k-CUT-SETS in hypergraphs for every fixed k (in addition to a polynomial-time algorithm to enumerate all of them for every fixed k). Our upper bound of $n^{O(k^2)}$ on the number of MINMAX-k-CUT-SETS is tight; there exist n-vertex connected graphs for which the number of MINMAX-k-CUT-SETS is $n^{\Theta(k^2)}$, where k depends on n (see Section 6).

Next, we briefly recall the status of HYPERGRAPH-k-CUT and ENUM-HYPERGRAPH-k-CUT. HYPERGRAPH-k-CUT was shown to be solvable in randomized polynomial time only recently (Chandrasekaran et al. [16], Fox et al. [24]); the randomized algorithms also show that the number of MIN-k-CUT-SETS is $O(n^{2k-2})$, and they can all be enumerated in randomized polynomial time. A subsequent deterministic algorithm was designed to solve HYPERGRAPH-k-CUT in time $n^{O(k)}p$ by Chandrasekaran and Chekuri [13]. Their techniques are extended to design the first deterministic polynomial-time algorithm to solve ENUM-HYPERGRAPH-k-CUT in Beideman et al. [7]. The algorithm for ENUM-HYPERGRAPH-k-CUT given in Beideman et al. [7] runs in time $n^{O(k^2)}p$. We note that this runtime has a quadratic dependence on k in the exponent of n although the number of MIN-k-CUT-SETS has only linear dependence on k in the exponent of n (because it is $O(n^{2k-2})$). So an open question that remained from Beideman et al. [7] is whether one can obtain an $n^{O(k)}p$-time deterministic algorithm for ENUM-HYPERGRAPH-k-CUT. We resolve this question affirmatively.

Theorem 2. *There exists a deterministic algorithm to solve* ENUM-HYPERGRAPH-K-CUT *that runs in time* $O(n^{16k-25}p)$, *where* n *is the number of vertices and* p *is the size of the input hypergraph.*

Our algorithms for both ENUM-MINMAX-HYPERGRAPH-k-PARTITION and ENUM-HYPERGRAPH-k-CUT are based on a structural theorem that allows for efficient recovery of optimum k-cut-sets via minimum (s, t)-terminal cuts (see

Theorem 5). Our structural theorem builds on structural theorems that appear in previous works on Hypergraph-k-Cut and MinMax-Hypergraph-k-Partition (Beideman et al. [7], Chandrasekaran and Chekuri [12, 13]). Our structural theorem may appear to be natural/incremental in comparison with ones that appear in previous works, but formalizing the theorem and proving it is a significant part of our contribution. Moreover, our single structural theorem is strong enough to enable efficient algorithms for both Enum-Hypergraph-k-Cut and Enum-MinMax-Hypergraph-k-Partition in contrast to previously known structural theorems. In this sense, our structural theorem can be viewed as a strong and unifying generalization of structural theorems that appear in previous works. We believe that our structural theorem is of independent interest in the theory of cuts and partitioning in hypergraphs (as well as graphs).

1.4. Technical Overview and Main Structural Result

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

1.4.1. Notation and Background. Let $G=(V,E)$ be a hypergraph. Throughout this work, n denotes the number of vertices in G, m denotes the number of hyperedges in G, and $p:=\sum_{e\in E}|e|$ denotes the representation size of G. We denote a partition of the vertex set into h nonempty parts by an ordered tuple $(V_1,\ldots,V_h)$ and call such an ordered tuple an h-partition. For a partition $\mathcal{P}=(V_1,V_2,\ldots,V_h)$, we say that a hyperedge e crosses the partition $\mathcal{P}$ if it intersects at least two parts of the partition. We refer to a 2-partition as a cut. For a nonempty proper subset U of vertices, we use $\overline{U}$ to denote $V\setminus U$, $\delta(U)$ to denote the set of hyperedges crossing the cut $(U,\overline{U})$, and $d(U):=|\delta(U)|$ to denote the cut value of U. We observe that $\delta(U)=\delta(\overline{U})$, so we use $d(U)$ to denote the value of the cut $(U,\overline{U})$. More generally, given a partition $\mathcal{P}=(V_1,V_2,\ldots,V_h)$, we denote the set of hyperedges crossing the partition by $\delta(V_1,V_2,\ldots,V_h)$ (also by $\delta(\mathcal{P})$ for brevity) and the number of hyperedges crossing the partition by $|\delta(V_1,V_2,\ldots,V_h)|$. We denote the optimum value of MinMax-Hypergraph-k-Partition and Hypergraph-k-Cut, respectively, by

$$OPT_{\text{minmax-}k\text{-partition}}:=\min\left\{\max_{i\in[k]}|\delta(V_i)|:(V_1,\ldots,V_k)\text{ is a }k\text{-partition of }V\right\}\text{ and}$$

$$OPT_{k\text{-cut}}:=\min\left\{|\delta(V_1,\ldots,V_k)|:(V_1,\ldots,V_k)\text{ is a }k\text{-partition of }V\right\}.$$

A key algorithmic tool is the use of fixed-terminal cuts. Let S, T be disjoint nonempty subsets of vertices. A 2-partition $(U,\overline{U})$ is an (S,T)-terminal cut if $S\subseteq U\subseteq V\setminus T$. Here, the set U is known as the source set, and the set $\overline{U}$ is known as the sink set. A minimum valued (S,T)-terminal cut is known as a minimum (S,T)-terminal cut. Because there can be multiple minimum (S,T)-terminal cuts, we are interested in source minimal minimum (S,T)-terminal cuts: a minimum (S,T)-terminal cut $(U,\overline{U})$ is a source minimal minimum (S,T)-terminal cut if there does not exist a minimum (S,T)-terminal cut $(U',\overline{U}')$ such that $U'\subsetneq U$. For every pair of disjoint nonempty subsets S and T of vertices, there exists a unique source minimal minimum (S,T)-terminal cut, and it can be found in deterministic polynomial time via standard maxflow algorithms. In particular, the source minimal minimum (S,T)-terminal cut can be found in time $O(np)$ (Chekuri and Xu [18]).

Our technique to enumerate all minmax-k-cut-sets and all min-k-cut-sets builds on the approaches of Chandrasekaran and Chekuri [12, 13] for Hypergraph-k-Cut and Minmax-SymSubmod-k-Partition (Beideman et al. [7]). We need the following structural theorem that is shown in Beideman et al. [7].

Theorem 3. (Beideman et al. [7]). *Let $G=(V,E)$ be a hypergraph and let $OPT_{k\text{-cut}}$ be the optimum value of* Hypergraph-k-Cut *in G for some integer $k\geq 2$. Suppose $(U,\overline{U})$ is a 2-partition of V with $d(U)<OPT_{k\text{-cut}}$. Then, for every pair of vertices $s\in U$ and $t\in\overline{U}$, there exist subsets $S\subseteq U\setminus\{s\}$ and $T\subseteq\overline{U}\setminus\{t\}$ with $|S|\leq 2k-3$ and $|T|\leq 2k-3$ such that $(U,\overline{U})$ is the unique minimum $(S\cup\{s\},T\cup\{t\})$-terminal cut in G.*

1.4.2. Enum-Hypergraph-k-Cut. We first focus on Enum-Hypergraph-k-Cut. We note that Theorem 3 allows us to recover those parts V_i of an optimum k-partition $(V_1,\ldots,V_k)$ for which $d(V_i)<OPT_{k\text{-cut}}$; the theorem tells us that each such V_i is the source side of a minimum (S,T)-cut for some $S,T\subseteq V$ such that $|S|,|T|\leq 2k-2$. However, recall that our goal is not to recover all optimum k-partitions for Hypergraph-k-Cut, but rather to recover all min-k-cut-sets (i.e., not to recover the parts of every optimum k-partition, but rather only to recover the k-cut set of every optimum k-partition). The previous work (Beideman et al. [7]) that designed an $n^{O(k^2)}p$-time deterministic enumeration algorithm achieved this by proving the following structural theorem.

Theorem 4. (Beideman et al. [7]). *Let $G = (V, E)$ be a hypergraph, $k \geq 2$ be an integer, and $\mathcal{P} = (V_1, \ldots, V_k)$ be an optimum k-partition for* Hypergraph-k-Cut *such that $\delta(V_1) = \delta(\mathcal{P})$. Then, for all subsets $T \subseteq \overline{V_1}$ such that $T \cap V_j \neq \emptyset$ for all $j \in [k] \setminus \{1\}$, there exists a subset $S \subseteq V_1$ with $|S| \leq 2k-1$ such that the source minimal minimum (S, T)-terminal cut $(A, \overline{A})$ satisfies $\delta(A) = \delta(V_1)$ and $A \subseteq V_1$.*

This structural theorem, in conjunction with Theorem 3, allows one to enumerate a candidate family $\mathcal{F}$ of $n^{O(k^2)}$ subsets of hyperedges such that every min-k-cut-set is present in the family. For every pair of disjoint subsets $S, T \subseteq V$ such that $|S|, |T| \leq 2k-1$, compute the source minimal minimum (S, T)-terminal cut $(A, \overline{A})$, and if $G - \delta(A)$ has at least k connected components, then add $\delta(A)$ to the candidate family $\mathcal{F}$; otherwise, add the source set A to a candidate collection $\mathcal{C}$. Next, for all possible k-partitions that can be formed using the sets in the collection $\mathcal{C}$, add the set of hyperedges crossing the k-partition to the family $\mathcal{F}$. The drawback of their structural theorem is that it is driven toward recovering the cut set $\delta(V_i)$ of every part V_i of every optimum k-partition $(V_1, \ldots, V_k)$. Hence, their algorithmic approach ends up with a runtime of $n^{O(k^2)}p$. In order to improve the runtime, we prove a stronger result: we show that for an arbitrary cut $(U, \overline{U})$ with cut value $OPT_{k\text{-cut}}$ (as opposed to only those sets V_i of an optimum k-partition $(V_1, \ldots, V_k)$), its cut set $\delta(U)$ can be recovered as the cut set of every minimum (S, T)-terminal cut for some S and T of small size. The following is the main structural theorem of this work.

Theorem 5. *Let $G = (V, E)$ be a hypergraph, and let $OPT_{k\text{-cut}}$ be the optimum value of* Hypergraph-k-Cut *in G for some integer $k \geq 2$. Suppose $(U, \overline{U})$ is a 2-partition of V with $d(U) = OPT_{k\text{-cut}}$. Then, there exist sets $S \subseteq U$, $T \subseteq \overline{U}$ with $|S| \leq 2k-1$ and $|T| \leq 2k-1$ such that every minimum (S, T)-terminal cut $(A, \overline{A})$ satisfies $\delta(A) = \delta(U)$.*

We encourage the reader to compare and contrast Theorems 3–5. Theorem 3 helps to recover cuts whose cut value is strictly smaller than $OPT_{k\text{-cut}}$, whereas Theorem 5 helps to recover cut sets whose size is equal to $OPT_{k\text{-cut}}$. Theorem 5 seems weaker because it only recovers cut sets, but its hypothesis is also weaker (we emphasize that recovering cut sets is the best possible that one can hope to do under its hypothesis as seen from the spanning hyperedge example). However, proving Theorem 5 requires us to work with cut sets (as opposed to cuts), which is a technical barrier to overcome. Indeed, our proof of Theorem 5 deviates significantly from the proof of Theorem 3 because we have to work with cut sets. We also note that our Theorem 5 is stronger than Theorem 4 on two fronts: (1) our result helps to recover the cut set $\delta(U)$ of an arbitrary cut $(U, \overline{U})$ whose cut value is $d(U) = OPT_{k\text{-cut}}$, whereas Theorem 4 helps only to recover the cut set $\delta(V_i)$ of a part V_i of an optimum k-partition $(V_1, \ldots, V_k)$ for Hypergraph-k-Cut whose cut value is $d(V_i) = OPT_{k\text{-cut}}$, and (2) our result does not need source minimality of the minimum (S, T)-terminal cuts in the conclusion. Moreover, the proof technique of Theorem 4 given in Beideman et al. [7] crucially relies on a containment property with respect to V_1 (namely, for every $S \subseteq V_1$ and every $T \subseteq \overline{V_1}$ such that $T \cap V_j \neq \emptyset$ for all $j \in [k] \setminus \{1\}$, the source minimal minimum (S, T)-terminal cut $(A, \overline{A})$ is such that $A \subseteq V_1$), whereas under the hypothesis of Theorem 5, the containment property fails with respect to the set U, and consequently, our proof technique differs from theirs.

Theorems 3 and 5 lead to a deterministic $n^{O(k)}$-time algorithm to enumerate all min-k-cut-sets via a divide-and-conquer approach. We describe this algorithm now. For each pair (S, T) of disjoint subsets of vertices S and T with $|S|, |T| \leq 2k-1$, compute the source minimal minimum (S, T)-terminal cut $(A, \overline{A})$; (i) if $G - \delta(A)$ has at least k connected components, then add $\delta(A)$ to the candidate family $\mathcal{F}$; (ii) otherwise, add the set A to a collection $\mathcal{C}$. We note that the sizes of the family $\mathcal{F}$ and the collection $\mathcal{C}$ are $O(n^{4k-2})$. Next, for each subset A in the collection $\mathcal{C}$, recursively enumerate all min-$k/2$-cut-sets in the subhypergraphs induced by A and $\overline{A}$ respectively[5]—denoted $G[A]$ and $G[\overline{A}]$, respectively—and add $\delta(A) \cup F_1 \cup F_2$ to the family $\mathcal{F}$ for each F_1 and F_2 being min-$k/2$-cut-set in $G[A]$ and $G[\overline{A}]$, respectively. Finally, return the subfamily of k-cut sets from the family $\mathcal{F}$ that are of smallest size.

We sketch the correctness analysis of this approach: let $F = \delta(V_1, \ldots, V_k)$ be a min-k-cut-set with $(V_1, \ldots, V_k)$ being an optimum k-partition for Hypergraph-k-Cut. We show that the family $\mathcal{F}$ contains F. Let $U := \cup_{i=1}^{k/2} V_i$. We note that $\delta(U) \subseteq F$. We have two possibilities. (1) Say $d(U) = |F|$. Then, $d(U) = OPT_{k\text{-cut}}$. Consequently, by Theorem 5, the min-k-cut-set F is added to the family $\mathcal{F}$ by step (i). (2) Say $d(U) < |F|$. Then, by Theorem 3, the set $U = \cup_{i=1}^{k/2} V_i$ is added to the collection $\mathcal{C}$ by step (ii); moreover, $F_1 := F \cap E(G[U])$ and $F_2 := F \cap E(G[\overline{U}])$ are min-$k/2$-cut-set in $G[U]$ and $G[\overline{U}]$, respectively, and they would have been enumerated by recursion, and hence, the set $\delta(U) \cup F_1 \cup F_2 = F$ is added to the family $\mathcal{F}$. The size of the family $\mathcal{F}$ can be shown to be $n^{O(k \log k)}$, and the runtime can be shown to be $n^{O(k \log k)}p$ (see Theorem 8). Using the known fact that the number of min-k-cut-sets in a n-vertex hypergraph is $O(n^{2k-2})$, we can improve the runtime analysis of this approach to $n^{O(k)}p$ (see Lemma 2).

1.4.3. Enum-MinMax-Hypergraph-k-Partition.

Next, we focus on Enum-MinMax-Hypergraph-k-Partition. There is a fundamental technical issue in enumerating minmax-k-cut-sets as opposed to min-k-cut-sets. We highlight this

technical issue now. Suppose we find an optimum k-partition $(V_1, \ldots, V_k)$ for MINMAX-HYPERGRAPH-k-PARTITION (say via Chandrasekaran and Chekuri's [12] algorithm) and store only the MINMAX-k-CUT-SET $F = \delta(V_1, \ldots, V_k)$ but forget to store the partition $(V_1, \ldots, V_k)$. Now, by knowing a MINMAX-k-CUT-SET F, can we recover some optimum k-partition for MINMAX-HYPERGRAPH-k-PARTITION (not necessarily $(V_1, \ldots, V_k)$)? Or, by knowing a MINMAX-k-CUT-SET F, is it even possible to find the value $OPT_{\text{minmax-}k\text{-partition}}$ without solving MINMAX-HYPERGRAPH-k-PARTITION from scratch again; that is, is there an advantage to knowing a MINMAX-k-CUT-SET in order to solve MINMAX-HYPERGRAPH-k-PARTITION? We are not aware of such an advantage. This is in stark contrast to HYPERGRAPH-k-CUT in which knowing a MIN-k-CUT-SET enables a linear-time solution to HYPERGRAPH-k-CUT.[6]

Why is this issue significant when solving ENUM-MINMAX-HYPERGRAPH-k-PARTITION? We recall that, in our approach for ENUM-HYPERGRAPH-k-CUT, the algorithm computed a polynomial-sized family $\mathcal{F}$ containing all MIN-k-CUT-SETS and returned the ones with smallest size; the smallest size ones are exactly MIN-k-CUT-SETS. It is unclear if a similar approach could work for enumerating MINMAX-k-CUT-SETS: suppose we do have an algorithm to enumerate a polynomial-sized family $\mathcal{F}$ containing all MINMAX-k-CUT-SETS. Now, in order to return all MINMAX-k-CUT-SETS (which is a subfamily of $\mathcal{F}$), note that we need to identify them among the ones in the family $\mathcal{F}$; that is, we need to verify if a given subset $F \in \mathcal{F}$ of hyperedges is a MINMAX-k-CUT-SET. This verification problem is closely related to the question mentioned in the previous paragraph. We do not know how to address this verification problem directly. So our algorithmic approach for ENUM-MINMAX-HYPERGRAPH-k-PARTITION has to overcome this technical issue.

Our ingredient to overcome this technical issue is to enumerate representatives for MINMAX-k-CUT-SETS. For a k-partition $(V_1, \ldots, V_k)$ and disjoint subsets $U_1, \ldots, U_k \subseteq V$, we call the k-tuple $(U_1, \ldots, U_k)$ a k-cut-set representative of $(V_1, \ldots, V_k)$ if $U_i \subseteq V_i$ and $\delta(U_i) = \delta(V_i)$ for all $i \in [k]$. We note that a fixed k-partition $(V_1, \ldots, V_k)$ could have several k-cut set representatives, and a fixed k-tuple $(U_1, \ldots, U_k)$ could be the k-cut set representative of several k-partitions. Yet it is possible to efficiently verify if a given k-tuple $(U_1, \ldots, U_k)$ is a k-cut set representative: consider the procedure that initializes $P_i = U_i$ for each $i \in [k]$ and for each connected component C in $G - \cup_{i=1}^{k} \delta(U_i)$. If $\delta(U_i) = \delta(U_i \cup C)$ for some $i \in [k]$, then the procedure adds C into P_i, and then, $(U_1, \ldots, U_k)$ is a k-cut set representative if and only if this procedure leads to a k-partition $P_1, \ldots, P_k$ of V (see Theorem 9). Moreover, knowing a k-cut set representative $(U_1, \ldots, U_k)$ of a k-partition $(V_1, V_2, \ldots, V_k)$ allows one to recover the k-cut set $F := \delta(V_1, \ldots, V_k)$ because $F = \cup_{i=1}^{k} \delta(U_i)$. Thus, in order to enumerate all MINMAX-k-CUT-SETS, it suffices to enumerate k-cut set representatives of all optimum k-partitions for MINMAX-HYPERGRAPH-k-PARTITION. At this point, the astute reader may wonder if there exists a polynomial-sized family of k-cut set representatives of all optimum k-partitions for MINMAX-HYPERGRAPH-k-PARTITION given that the number of optimum k-partitions for MINMAX-HYPERGRAPH-k-PARTITION could be exponential. For example, is there a polynomial-sized family of k-cut set representatives of all optimum k-partitions for MINMAX-HYPERGRAPH-k-PARTITION in the spanning hyperedge example? Indeed, in the spanning hyperedge example, even though the number of optimum k-partitions for MINMAX-HYPERGRAPH-k-PARTITION is exponential, there exists a $\left(k!\binom{n}{k}\right)$-sized family of k-cut set representatives of all optimum k-partitions: consider the family $\{(\{v_1\}, \ldots, \{v_k\}) : v_1, \ldots, v_k \in V, v_i \neq v_j \ \forall \text{distinct } i, j \in [k]\}$.

It turns out that Theorems 3 and 5 are strong enough to enable efficient enumeration of k-cut set representatives of all optimum k-partitions for MINMAX-HYPERGRAPH-k-PARTITION. We describe the algorithm to achieve this. For each pair (S, T) of disjoint subsets of vertices with $|S|, |T| \leq 2k - 1$, compute the source minimal minimum (S, T)-terminal cut $(U, \overline{U})$ and add U to a candidate collection $\mathcal{C}$. We note that the size of the collection $\mathcal{C}$ is $O(n^{4k-2})$. Next, for each k-tuple $(U_1, \ldots, U_k) \in \mathcal{C}^k$, verify if $(U_1, \ldots, U_k)$ is a k-cut set representative (using Theorem 9), and if so, then add the k-tuple to the candidate family $\mathcal{D}$. Finally, return $\arg\min\{\max_{i=1}^{k} d(U_i) : (U_1, \ldots, U_k) \in \mathcal{D}\}$; that is, prune and return the subfamily of k-cut set representatives $(U_1, \ldots, U_k)$ from the family $\mathcal{D}$ that have minimum $\max_{i=1}^{k} d(U_i)$.

We note that the size of the family $\mathcal{D}$ is $n^{O(k^2)}$, and consequently, the runtime is $n^{O(k^2)}p$. We sketch the correctness analysis of this approach. Let $(V_1, \ldots, V_k)$ be an optimum k-partition for MINMAX-HYPERGRAPH-k-PARTITION. We show that the family $\mathcal{D}$ contains a k-cut set representative of $(V_1, \ldots, V_k)$. By noting that $OPT_{\text{minmax-}k\text{-partition}} \leq OPT_{k\text{-cut}}$ and by Theorems 3 and 5, for every $i \in [k]$, we have a set U_i in the collection $\mathcal{C}$ with $U_i \subseteq V_i$ and $\delta(U_i) = \delta(V_i)$. Hence, the k-tuple $(U_1, \ldots, U_k) \in \mathcal{C}^k$ is a k-cut set representative, and it is added to the family $\mathcal{D}$. The final pruning step does not remove $(U_1, \ldots, U_k)$ from the family $\mathcal{D}$, and hence, it is in the subfamily returned by the algorithm.

1.4.4. Significance of Our Technique. As mentioned earlier, our techniques build on the structural theorems that appear in previous works (Beideman et al. [7], Chandrasekaran and Chekuri [12, 13]). The main technical novelty of our contribution lies in Theorem 5, which can be viewed as the culmination of structural theorems developed in those previous works. We also emphasize that using minimum (s, t)-terminal cuts to solve global partitioning problems is not a new technique per se (e.g., minimum (s, t)-terminal cut is the first and most natural approach to solve global minimum cut). This technique of using minimum (s, t)-terminal cuts to solve global partitioning problems

has a rich variety of applications in combinatorial optimization: for example, (1) it is used to design the first efficient algorithm for Graph-k-Cut for fixed k (Goldschmidt and Hochbaum [27]), (2) it is used to design efficient algorithms for certain constrained submodular minimization problems (Goemans and Ramakrishnan [26], Nägele et al. [51]), and (3) more recently, it was used to design fast algorithms for global minimum cut in graphs as well as to obtain fast Gomory–Hu trees in unweighted graphs (Abboud et al. [1], Li and Panigrahi [46]). The applicability of this technique relies on identifying and proving appropriate structural results. Our Theorem 5 is such a structural result. The merit of the structural result lies in its ability to solve two different enumeration problems in hypergraph k-partitioning which was not possible via structural theorems that were developed before. Moreover, it leads to the first polynomial bound on the number of minmax-k-cut-sets in hypergraphs for every fixed k.

1.4.5. Organization. We discuss related work in Section 1.5. In Section 1.6, we recall properties of the hypergraph cut function. In Section 2, we prove a special case of Theorem 5. In Section 3, we use this special case to prove Theorem 5. In Section 4, we design an efficient algorithm for Enum-Hypergraph-k-Cut and prove Theorem 2. In Section 5, we design an efficient algorithm for Enum-MinMax-Hypergraph-k-Partition and prove Theorem 1. We present a lower bound example in Section 6. We conclude with an open question in Section 7.

1.5. Related Work

We briefly discuss the status of the enumeration problems for $k = 2$ followed by the status for $k \geq 2$ in graphs and hypergraphs.

1.5.1. Enumeration Problems for $k = 2$. For $k = 2$, both Enum-MinMax-Hypergraph-k-Partition and Enum-Hypergraph-k-Cut are equivalent to enumerating optimum solutions for global minimum cut in hypergraphs. For graphs that are connected, it is well-known that the number of minimum cuts and of α-approximate minimum cuts are at most $\binom{n}{2}$ and $O(n^{2\alpha})$, respectively, and they can all be enumerated in polynomial time for constant α (Chekuri et al. [19], Dinitz et al. [22], Goemans and Ramakrishnan [26], Henzinger and Williamson [34], Karger [38], Nagamochi et al. [50]). For connected hypergraphs, the number of minimum cuts can be exponential as seen from the spanning hyperedge example. However, recent results show that the number of minimum cut sets in a hypergraph is at most $\binom{n}{2}$ via several different techniques, and they can all be enumerated in polynomial time (Beideman et al. [7], Chandrasekaran et al. [16], Chekuri and Xu [18], Fox et al. [24], Ghaffari et al. [25]). On the other hand, there exist hypergraphs with an exponential number of 2-approximate minimum cut sets.[7]

1.5.2. Graph-k-Cut and Enum-Graph-k-Cut. When k is part of the input, Graph-k-Cut is NP-hard (Goldschmidt and Hochbaum [27]) and admits a $2(1 - 1/k)$-approximation (Gupta et al. [30], Quanrud [53], Ravi and Sinha [56], Saran and Vazirani [57], Zhao et al. [63]). Manurangsi [49] shows that there is no polynomial-time $(2 - \epsilon)$-approximation for any constant $\epsilon > 0$ assuming the small set expansion hypothesis (Raghavendra and Steurer [55]). We note that Graph-k-Cut is $W[1]$-hard when parameterized by k (Downey et al. [23]) and admits a fixed-parameter approximation scheme when parameterized by k (Gupta et al. [29, 30], Kawarabayashi and Lin [42], Lokshtanov et al. [47]) and is fixed-parameter tractable when parameterized by k and the solution size (Kawarabayashi and Thorup [43]).

Graph-k-Cut for fixed k is shown to be polynomial-time solvable by Goldschmidt and Hochbaum [27]. Subsequently, Karger and Stein [40] give a randomized polynomial-time algorithm whose analysis also shows that the number of optimum k-partitions in a connected graph is $O(n^{2k-2})$ and they can all be enumerated in polynomial time for every fixed k (also see Chekuri et al. [19], Kamidoi et al. [36], Thorup [59], Xiao [60]). The number of optimum k-partitions has recently been improved to $O(n^k)$ for fixed k, thereby leading to a faster algorithm for Graph-k-Cut for fixed k (Gupta et al. [31–33]).

1.5.3. Hypergraph-k-Cut and Enum-Hypergraph-k-Cut. When k is part of input, Hypergraph-k-Cut is at least as hard as the densest k-subgraph problem (Chekuri and Li [17]). Combined with results in Manurangsi [48], this implies that Hypergraph-k-Cut is unlikely to have a subpolynomial factor approximation ratio. Moreover, Hypergraph-k-Cut is $W[1]$-hard even when parameterized by k and the solution size (see Chandrasekaran and Chekuri [13]). These two hardness results illustrate that Hypergraph-k-Cut differs significantly from Graph-k-Cut in complexity.

Hypergraph-k-Cut for fixed k is recently shown to be polynomial-time solvable via a randomized algorithm (Chandrasekaran et al. [16], Fox et al. [24]). The analysis of the randomized algorithm also shows that the number of min-k-cut-sets is $O(n^{2k-2})$, and they can all be enumerated in randomized polynomial time. A deterministic

polynomial-time algorithm was given by Chandrasekaran and Chekuri [13]. Subsequently, a deterministic polynomial-time algorithm for Enum-Hypergraph-k-Cut for fixed k was given by Beideman et al. [7].

1.5.4. Minmax-Graph-k-Partition and Enum-MinMax-Graph-k-Partition. When k is part of input, Minmax-Graph-k-Partition is NP-hard (Chandrasekaran and Wang [14]), whereas its approximability is open; we do not yet know if it admits a constant factor approximation. When parameterized by k, it is $W[1]$-hard and admits a fixed-parameter approximation scheme (Chandrasekaran and Wang [14]).

Minmax-Graph-k-Partition for fixed k is polynomial-time solvable via the following observation (see Chandrasekaran and Chekuri [12], Chandrasekaran and Wang [14]): in connected graphs, an optimum k-partition for Minmax-Graph-k-Partition is a k-approximate solution to Graph-k-Cut. The randomized algorithm of Karger and Stein [40] implies that the number of k-approximate solutions to Graph-k-Cut is $n^{O(k^2)}$, and they can all be enumerated in polynomial time (Chekuri et al. [19], Gupta et al. [32, 33], Karger and Stein [40]). These two facts together imply that Minmax-Graph-k-Partition can be solved in time $n^{O(k^2)}$, and moreover, the number of optimum k-partitions for Minmax-Graph-k-Partition in a connected graph is $n^{O(k^2)}$, and they can all be enumerated in polynomial time for constant k.

1.5.5. Minmax-Hypergraph-k-Partition and Enum-MinMax-Hypergraph-k-Partition. Minmax-Hypergraph-k-Partition is discussed as early as 1973 by Lawler [45]. When k is part of input, Minmax-Hypergraph-k-Partition is at least as hard as the densest k-subgraph problem (this follows from the reduction in Chekuri and Li [17] and was observed by Chandrasekaran and Zhang [15]). For fixed k, the approach for Minmax-Graph-k-Partition described does not extend to Minmax-Hypergraph-k-Partition. This is because the number of k-approximate solutions to Hypergraph-k-Cut can be exponential, and hence, they cannot be enumerated efficiently (e.g., we have already seen that the number of 2-approximate minimum cut sets in a hypergraph can be exponential). Chandrasekaran and Chekuri [12] give a deterministic polynomial-time algorithm for the more general problem of Minmax-SymSubmod-k-Partition for fixed k, which, in turn, implies that Minmax-Hypergraph-k-Partition is also solvable efficiently for fixed k. Their algorithm finds an optimum k-partition for Minmax-Hypergraph-k-Partition and is not conducive to enumerate all minmax-k-cut-sets. We emphasize that no polynomial bound on the number of minmax-k-cut-sets for fixed k was known prior to our work.

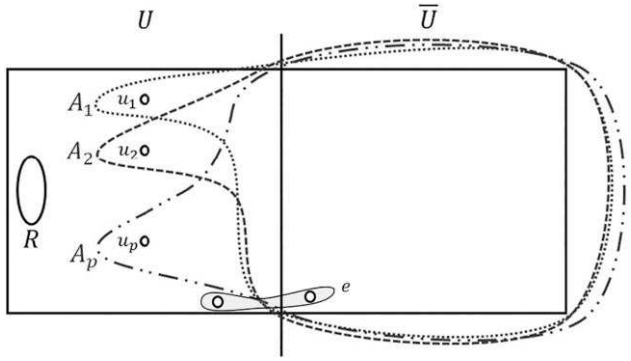
For a detailed discussion on other hypergraph k-partitioning problems that are special cases of Minsum-Submod-k-Partition, we refer the reader to Zhao et al. [63], Zhao [62], and Okumoto et al. [52].

1.6. Preliminaries

Let $G = (V, E)$ be a hypergraph. Throughout, we follow the notation mentioned in the second paragraph of Section 1.4. For disjoint $A, B \subseteq V$, we define $E(A, B) := \{e \in E : e \subseteq A \cup B, e \cap A \neq \emptyset, e \cap B \neq \emptyset\}$, and $E[A] := \{e \in E : e \subseteq A\}$. We repeatedly rely on the fact that the hypergraph cut function $d : 2^V \to \mathbb{R}_+$ is symmetric and submodular. We recall that a set function $f : 2^V \to \mathbb{R}$ is symmetric if $f(U) = f(\overline{U})$ for all subsets $U \subseteq V$ and is submodular if $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$ for all subsets $A, B \subseteq V$.

We need the following theorem that is proved in previous works on Hypergraph-k-Cut and Enum-Hypergraph-k-Cut (see Figure 1 for an illustration of the sets that appear in the statement of Theorem 6).

Figure 1. Illustration of the sets that appear in the statement of Theorem 6. A hyperedge e of the form in the second part of the statement is also shown.



Theorem 6 (Beideman et al. [7], Chandrasekaran and Chekuri [13]). *Let $G = (V, E)$ be a hypergraph, $k \geq 2$ be an integer, and $\emptyset \neq R \subsetneq U \subsetneq V$. Let $S = \{u_1, \ldots, u_p\} \subseteq U \setminus R$ for $p \geq 2k-2$. Let $(\overline{A_i}, A_i)$ be a minimum $((S \cup R) \setminus \{u_i\}, \overline{U})$-terminal cut. Suppose that $u_i \in A_i \setminus (\cup_{j \in [p] \setminus \{i\}} A_j)$ for every $i \in [p]$. Then, the following two hold:*

1. *There exists a k-partition $(P_1, \ldots, P_k)$ of V with $\overline{U} \subsetneq P_k$ such that*

$$|\delta(P_1, \ldots, P_k)| \leq \frac{1}{2} \min\{d(A_i) + d(A_j) : i, j \in [p], i \neq j\}.$$

2. *Moreover, if there exists a hyperedge $e \in E$ such that e intersects $W := \cup_{1 \leq i < j \leq p} (A_i \cap A_j)$, e intersects $Z := \cap_{i \in [p]} \overline{A_i}$, and e is contained in $W \cup Z$, then the inequality in the previous conclusion is strict.*

We use Theorem 6 in our proof of Theorem 5, which proceeds via contradiction. Theorem 6 allows us to construct a k-partition with number of crossing hyperedges being strictly less than the optimum value for Hypergraph-k-Cut, thus leading to a contradiction.

2. A Special Case of Theorem 5

Theorem 7 is the main theorem of this section. Theorem 7 implies Theorem 5 in the special case in which the 2-partition $(U, \overline{U})$ of interest to Theorem 5 is such that $|\overline{U}| \leq 2k-1$. We encourage readers interested in understanding the motivation for considering this special case to read the statement of Theorem 7, jump ahead to Section 3 in which we prove Theorem 5 using Theorem 7, and then return to the proof of Theorem 7.

Theorem 7. *Let $G = (V, E)$ be a hypergraph, and let $OPT_{k\text{-cut}}$ be the optimum value of* Hypergraph-k-Cut *in G for some integer $k \geq 2$. Suppose $(U, \overline{U})$ is a 2-partition of V with $d(U) = OPT_{k\text{-cut}}$. Then, there exists a set $S \subseteq U$ with $|S| \leq 2k-1$ such that every minimum $(S, \overline{U})$-terminal cut $(A, \overline{A})$ satisfies $\delta(A) = \delta(U)$.*

Proof. Consider the collection

$$\mathcal{C} := \{Q \subseteq V : \overline{U} \subsetneq Q,\ d(Q) \leq d(U),\ \text{and } \delta(Q) \neq \delta(U)\}.$$

Let S be an inclusion-wise minimal subset of U such that $S \cap Q \neq \emptyset$ for all $Q \in \mathcal{C}$; that is, the set S is completely contained in U and is a minimal transversal of the collection $\mathcal{C}$. Proposition 1 and Lemma 1 complete the proof of Theorem 7 for this choice of S. □

Proposition 1. *Every minimum $(S, \overline{U})$-terminal cut $(A, \overline{A})$ has $\delta(A) = \delta(U)$.*

Proof. Let $(A, \overline{A})$ be a minimum $(S, \overline{U})$-terminal cut. If $A = U$, then we are done, so we may assume that $A \neq U$. This implies that $S \subseteq A$ and $\overline{U} \subsetneq \overline{A}$. Because $(U, \overline{U})$ is a $(S, \overline{U})$-terminal cut, we have that $d(\overline{A}) = d(A) \leq d(U)$. Because S intersects every set in the collection $\mathcal{C}$, we have that $\overline{A} \notin \mathcal{C}$. Hence, $\delta(\overline{A}) = \delta(U)$, and by symmetry of cut sets, $\delta(A) = \delta(U)$. □

Lemma 1. *The size of the subset S is at most $2k-1$.*

Proof. For the sake of contradiction, suppose $|S| \geq 2k$. Our proof strategy is to show the existence of a k-partition with fewer crossing hyperedges than $OPT_{k\text{-cut}}$, thus contradicting the definition of $OPT_{k\text{-cut}}$. Let $S := \{u_1, u_2, \ldots, u_p\}$ for some $p \geq 2k$. For notational convenience, we use $S - u_i$ to denote $S \setminus \{u_i\}$ and $S - u_i - u_j$ to denote $S \setminus \{u_i, u_j\}$. For a subset $X \subseteq U$, we denote the source minimal minimum $(X, \overline{U})$-terminal cut by $(H_X, \overline{H_X})$.

Our strategy to arrive at a k-partition with fewer crossing hyperedges than $OPT_{k\text{-cut}}$ is to apply the second conclusion of Theorem 6. The next few claims set us up to obtain sets that satisfy the hypothesis of Theorem 6. We emphasize that we apply Theorem 6 twice: first, in Claim 3 and, second, to prove the existence of a k-partition with fewer than $OPT_{k\text{-cut}}$ crossing hyperedges.

Claim 1. *For every $i \in [p]$, we have $\overline{H_{S-u_i}} \in \mathcal{C}$.*

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

$$d(H_{S-u_i}) = d(\overline{H_{S-u_i}}) \leq d(B_i) \leq d(U).$$

Because $(H_{S-u_i}, \overline{H_{S-u_i}})$ is a $(S - u_i, \overline{U})$-terminal cut, we have that $\overline{U} \subseteq \overline{H_{S-u_i}}$. If $d(\overline{H_{S-u_i}}) < d(U)$, then $\delta(\overline{H_{S-u_i}}) \neq \delta(U)$ and $\overline{U} \subsetneq \overline{H_{S-u_i}}$, and consequently, $\overline{H_{S-u_i}} \in \mathcal{C}$. So we assume henceforth that $d(\overline{H_{S-u_i}}) = d(U)$.

Because $(H_{S-u_i} \cap \overline{B_i}, \overline{H_{S-u_i} \cap \overline{B_i}})$ is a $(S-u_i, \overline{U})$-terminal cut and $(H_{S-u_i}, \overline{H_{S-u_i}})$ is a minimum $(S-u_i, \overline{U})$-terminal cut, we have that

$$d(H_{S-u_i} \cap \overline{B_i}) \geq d(H_{S-u_i}).$$

Because $(H_{S-u_i} \cup \overline{B_i}, \overline{H_{S-u_i} \cup \overline{B_i}})$ is a $(S-u_i, \overline{U})$-terminal cut and $(H_{S-u_i}, \overline{H_{S-u_i}})$ is a minimum $(S-u_i, \overline{U})$-terminal cut, we have that

$$d(H_{S-u_i} \cup \overline{B_i}) \geq d(H_{S-u_i}).$$

Therefore, by submodularity of the hypergraph cut function, we have that

$$2d(U) \geq d(H_{S-u_i}) + d(B_i) \geq d(H_{S-u_i} \cap \overline{B_i}) + d(H_{S-u_i} \cup \overline{B_i}) \geq 2d(H_{S-u_i}) = 2d(U). \tag{1}$$

Therefore, all these inequalities should be equations. In particular, we have that $d(H_{S-u_i} \cap \overline{B_i}) = d(U) = d(B_i) = d(H_{S-u_i})$, and hence, $(H_{S-u_i} \cap \overline{B_i}, \overline{H_{S-u_i} \cap \overline{B_i}})$ is a minimum $(S-u_i, \overline{U})$-terminal cut. Because $(H_{S-u_i}, \overline{H_{S-u_i}})$ is a source minimal minimum $(S-u_i, \overline{U})$-terminal cut, we must have $H_{S-u_i} \cap \overline{B_i} = H_{S-u_i}$, and thus, $H_{S-u_i} \subseteq \overline{B_i}$. Therefore, $B_i \subseteq \overline{H_{S-u_i}}$. Because $B_i \in \mathcal{C}$, we have $\delta(B_i) \neq \delta(U)$. However, $d(B_i) = d(U)$. Therefore, $\delta(U) \setminus \delta(B_i) \neq \emptyset$. Let $e \in \delta(U) \setminus \delta(B_i)$. Because $e \in \delta(\overline{U})$, but $e \notin \delta(B_i)$, and $\overline{U} \subseteq B_i$, we have that $e \subseteq B_i$, and thus, $e \subseteq \overline{H_{S-u_i}}$. Thus, we conclude that $\delta(U) \setminus \delta(\overline{H_{S-u_i}}) \neq \emptyset$, and so $\delta(\overline{H_{S-u_i}}) \neq \delta(U)$. This also implies that $\overline{U} \subsetneq \overline{H_{S-u_i}}$. Thus, $\overline{H_{S-u_i}} \in \mathcal{C}$. □

Claim 1 implies the following corollary.

Corollary 1. *For every $i \in [p]$, we have $u_i \in \overline{H_{S-u_i}}$.*

Proof. By definition, $S - u_i \subseteq H_{S-u_i}$, so $S \cap \overline{H_{S-u_i}} \subseteq \{u_i\}$. By Claim 1, we have that $\overline{H_{S-u_i}} \in \mathcal{C}$. Because S is a transversal of the collection $\mathcal{C}$, we have that $S \cap \overline{H_{S-u_i}} \neq \emptyset$. So the vertex u_i must be in $\overline{H_{S-u_i}}$. □

Having obtained Corollary 1, the next few claims (Claims 2–5) are similar to the claims appearing in the proof of Theorem 4 that appear in Beideman et al. [7, claims 4.2–4.5 in the proof of theorem 1.3]). Because the hypothesis of the structural theorem that we are proving here is different from theirs, we present the complete proofs of these claims here. The way in which we use the claims is also different from Beideman et al. [7].

The following claim helps in showing that $u_i, u_j \notin H_{S-u_i-u_j}$, which, in turn, is used to show that the hypothesis of Theorem 6 is satisfied by suitably chosen sets.

Claim 2. *For every $i, j \in [p]$, we have $H_{S-u_i-u_j} \subseteq H_{S-u_i}$.*

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

$$d(H_{S-u_i-u_j} \cap H_{S-u_i}) \geq d(H_{S-u_i-u_j}). \tag{2}$$

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

$$d(H_{S-u_i-u_j} \cup H_{S-u_i}) \geq d(H_{S-u_i}). \tag{3}$$

By submodularity of the hypergraph cut function and Inequalities (2) and (3), we have that

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

Therefore, Inequality (2) is an equation, and consequently, $(H_{S-u_i-u_j} \cap H_{S-u_i}, \overline{H_{S-u_i-u_j} \cap H_{S-u_i}})$ is a minimum $(S-u_i-u_j, \overline{U})$-terminal cut. If $H_{S-u_i-u_j} \setminus H_{S-u_i} \neq \emptyset$, then $(H_{S-u_i-u_j} \cap H_{S-u_i}, \overline{H_{S-u_i-u_j} \cap H_{S-u_i}})$ contradicts source minimality of the minimum $(S-u_i-u_j, \overline{U})$-terminal cut $(H_{S-u_i-u_j}, \overline{H_{S-u_i-u_j}})$. Hence, $H_{S-u_i-u_j} \setminus H_{S-u_i} = \emptyset$, and consequently, $H_{S-u_i-u_j} \subseteq H_{S-u_i}$. □

Claim 2 implies the following corollary.

Corollary 2. *For every $i, j \in [p]$, we have $u_i, u_j \notin H_{S-u_i-u_j}$.*

Proof. By Corollary 1, we have that $u_i \notin H_{S-u_i}$. Therefore, $u_i, u_j \notin H_{S-u_i} \cap H_{S-u_j}$. By Claim 2, $H_{S-u_i-u_j} \subseteq H_{S-u_i}$ and $H_{S-u_i-u_j} \subseteq H_{S-u_j}$. Therefore, $H_{S-u_i-u_j} \subseteq H_{S-u_i} \cap H_{S-u_j}$, and thus, $u_i, u_j \notin H_{S-u_i-u_j}$. □

The next claim shows that the cut values $d(H_{S-u_i})$ and $d(H_{S-u_i-u_j})$ for every $i, j \in [p]$ are equal to $d(U)$.

Claim 3. *For every $i, j \in [p]$, we have $d(H_{S-u_i}) = d(U) = d(H_{S-u_i-u_j})$.*

Proof. Let $a, b \in [p]$. We show that $d(H_{S-u_a}) = d(U) = d(H_{S-u_a-u_b})$. Because $(U, \overline{U})$ is a $(S-u_a, \overline{U})$-terminal cut, we have that $d(H_{S-u_a}) \le d(U)$. Because $(H_{S-u_a}, \overline{H_{S-u_a}})$ is a $(S-u_a-u_b, \overline{U})$-terminal cut, we have that $d(H_{S-u_a-u_b}) \le d(H_{S-u_a}) \le d(U)$. Thus, in order to prove the claim, it suffices to show that $d(H_{S-u_a-u_b}) \ge d(U)$.

Suppose for contradiction that $d(H_{S-u_a-u_b}) < d(U)$. Let $\ell \in [p] \setminus \{a, b\}$ be an arbitrary element (which exists because we assume that $p \ge 2k$ and $k \ge 2$). Let $R := \{u_\ell\}$, $S' := S - u_a - u_\ell$, and $A_i := \overline{H_{S-u_a-u_i}}$ for every $i \in [p] \setminus \{a, \ell\}$. We note that $|S'| = p - 2 \ge 2k - 2$. By definition, $(\overline{A_i}, A_i)$ is a minimum $(S-u_a-u_i, \overline{U})$-terminal cut for every $i \in [p] \setminus \{a, \ell\}$. Moreover, for every $i \in [p] \setminus \{a, \ell\}$, we have $u_i \in \overline{H_{S-u_a-u_i}} = A_i$ by Corollary 2, and for every $j \in [p] \setminus \{a, i, \ell\}$, we have $u_i \in S - u_a - u_j \subseteq H_{S-u_a-u_j} = \overline{A_j}$. These imply that $u_i \in A_i \setminus (\cup_{j \in [p] \setminus \{a,i,\ell\}} A_j)$ for every $i \in [p] \setminus \{a, \ell\}$. Hence, the sets U, R, and S' and the cuts $(\overline{A_i}, A_i)$ for $i \in [p] \setminus \{a, \ell\}$ satisfy the conditions of Theorem 6. Therefore, by the first conclusion of Theorem 6, there exists a k-partition $\mathcal{P}'$ with

$$|\delta(\mathcal{P}')| \le \frac{1}{2} \min\{d(H_{S-u_a-u_i}) + d(H_{S-u_a-u_j}) : i, j \in [p] \setminus \{a, \ell\}\}.$$

By assumption, $d(H_{S-u_a-u_b}) < d(U)$ and $b \in [p] \setminus \{a, \ell\}$, so $\min\{d(H_{S-u_a-u_i}) : i \in [p] \setminus \{a, \ell\}\} < d(U)$. Because $(U, \overline{U})$ is a $(S-u_a-u_i, \overline{U})$-terminal cut, we have that $d(H_{S-u_a-u_i}) \le d(U)$ for every $i \in [p] \setminus \{a, \ell\}$. Therefore,

$$\frac{1}{2} \min\{d(H_{S-u_a-u_i}) + d(H_{S-u_a-u_j}) : i, j \in [p] \setminus \{a, \ell\}\} < d(U) = OPT_{k\text{-cut}}.$$

Thus, we have that $|\delta(\mathcal{P}')| < OPT_{k\text{-cut}}$, which is a contradiction. □

The next two claims help in arguing the existence of a hyperedge satisfying the conditions of the second conclusion of Theorem 6. In particular, we need Claim 5. The following claim helps in proving Claim 5.

Claim 4. *For every $i, j \in [p]$, we have*

$$d(H_{S-u_i} \cap H_{S-u_j}) = d(U) = d(H_{S-u_i} \cup H_{S-u_j}).$$

Proof. Because $(H_{S-u_i} \cap H_{S-u_j}, \overline{H_{S-u_i} \cap H_{S-u_j}})$ is a $(S-u_i-u_j, \overline{U})$-terminal cut, we have that $d(H_{S-u_i} \cap H_{S-u_j}) \ge d(H_{S-u_i-u_j})$. By Claim 3, we have that $d(H_{S-u_i-u_j}) = d(U) = d(H_{S-u_i})$. Therefore,

$$d(H_{S-u_i} \cap H_{S-u_j}) \ge d(H_{S-u_i}). \tag{4}$$

Because $(H_{S-u_i} \cup H_{S-u_j}, \overline{H_{S-u_i} \cup H_{S-u_j}})$ is a $(S-u_j, \overline{U})$-terminal cut, we have that

$$d(H_{S-u_i} \cup H_{S-u_j}) \ge d(H_{S-u_j}). \tag{5}$$

By submodularity of the hypergraph cut function and Inequalities (4) and (5), we have that

$$d(H_{S-u_i}) + d(H_{S-u_j}) \ge d(H_{S-u_i} \cap H_{S-u_j}) + d(H_{S-u_i} \cup H_{S-u_j}) \ge d(H_{S-u_i}) + d(H_{S-u_j}).$$

Therefore, Inequalities (4) and (5) are equations. Thus, by Claim 3, we have that

$$d(H_{S-u_i} \cap H_{S-u_j}) = d(H_{S-u_i}) = d(U),$$

and

$$d(H_{S-u_i} \cup H_{S-u_j}) = d(H_{S-u_j}) = d(U). \quad \square$$

Claim 5. *For every $i, j, \ell \in [p]$ with $i \ne j$, we have $H_{S-u_\ell} \subseteq H_{S-u_i} \cup H_{S-u_j}$.*

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

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

$$d(H_{S-u_\ell} \cap (H_{S-u_i} \cup H_{S-u_j})) = d(H_{S-u_\ell} \setminus Q) \ge d(H_{S-u_\ell}). \tag{6}$$

We also have that $(H_{S-u_\ell} \cup (H_{S-u_i} \cup H_{S-u_j}), \overline{H_{S-u_\ell} \cup (H_{S-u_i} \cup H_{S-u_j})})$ is a $(S-u_i, \overline{U})$-terminal cut. Therefore, $d(H_{S-u_\ell} \cup (H_{S-u_i} \cup H_{S-u_j})) \geq d(H_{S-u_i})$. By Claims 3 and 4, we have that $d(H_{S-u_i}) = d(U) = d(H_{S-u_i} \cup H_{S-u_j})$. Therefore,

$$d(H_{S-u_\ell} \cup (H_{S-u_i} \cup H_{S-u_j})) \geq d(H_{S-u_i} \cup H_{S-u_j}). \tag{7}$$

By submodularity of the hypergraph cut function and Inequalities (6) and (7), we have that

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

Therefore, Inequalities (6) and (7) are equations, so $(H_{S-u_\ell} \setminus Q, \overline{H_{S-u_\ell} \setminus Q})$ is a minimum $(S-u_\ell, \overline{U})$-terminal cut. □

Let $R := \{u_p\}$, $S' := S - u_p$, and $(\overline{A_i}, A_i) := (H_{S-u_i}, \overline{H_{S-u_i}})$ for every $i \in [p-1]$. By definition, $(\overline{A_i}, A_i)$ is a minimum $(S-u_i, \overline{U})$-terminal cut for every $i \in [p-1]$. Moreover, by Corollary 1, we have that $u_i \in A_i \setminus (\cup_{j \in [p-1] \setminus \{i\}} A_j)$. Hence, the sets U, R, and S' and the cuts $(\overline{A_i}, A_i)$ for $i \in [p-1]$ satisfy the conditions of Theorem 6. We now show that there exists a hyperedge satisfying the conditions mentioned in the second conclusion of Theorem 6. We use Claim 6 to prove this. Let $W := \cup_{1 \leq i < j \leq p-1} (A_i \cap A_j)$ and $Z := \cap_{i \in [p-1]} \overline{A_i}$ as in the statement of Theorem 6.

Claim 6. *There exists a hyperedge $e \in E$ such that $e \cap W \neq \emptyset$, $e \cap Z \neq \emptyset$, and $e \subseteq W \cup Z$.*

Proof. We note that $S \subseteq (S-u_i) \cup (S-u_j) \subseteq H_{S-u_i} \cup H_{S-u_j}$ for every distinct $i, j \in [p-1]$. Therefore, $S \cap (A_i \cap A_j) = \emptyset$ for every distinct $i, j \in [p-1]$, and thus, $S \cap W = \emptyset$. Because S is a transversal of the collection $\mathcal{C}$, it follows that the set W is not in the collection $\mathcal{C}$.

By definition, $\overline{U} \subseteq A_i$ for every $i \in [p-1]$, and thus, $\overline{U} \subseteq W$. Because $W \notin \mathcal{C}$, by definition of $\mathcal{C}$, we have that either $W = \overline{U}$ or $d(W) > d(U)$ or $\delta(W) = \delta(U)$. Because $W = \overline{U}$ is a special case of $\delta(W) = \delta(U)$, we have that either $d(W) > d(U)$ or $\delta(W) = \delta(U)$. By Claim 1, we have that $\overline{H_{S-u_p}} \in \mathcal{C}$, and thus, $d(\overline{H_{S-u_p}}) \leq d(U)$ and $\delta(\overline{H_{S-u_p}}) \neq \delta(U)$. Consequently, $d(W) \geq d(U) \geq d(\overline{H_{S-u_p}})$, and $\delta(W) = \delta(U) \neq \delta(\overline{H_{S-u_p}})$, and thus, $\delta(W) \setminus \delta(\overline{H_{S-u_p}}) \neq \emptyset$. Let $e \in \delta(W) \setminus \delta(\overline{H_{S-u_p}})$. We show that this choice of e achieves the desired properties.

For each $i \in [p]$, let $Y_i := \overline{H_{S-u_i}} \setminus W$. By Claim 5, for every $i, j, \ell \in [p]$ with $i \neq j$, we have that $H_{S-u_\ell} \subseteq H_{S-u_i} \cup H_{S-u_j}$. Therefore $\overline{H_{S-u_i}} \cap \overline{H_{S-u_j}} \subseteq \overline{H_{S-u_\ell}}$ for every such $i, j, \ell \in [p]$, and hence, $W \subseteq \overline{H_{S-u_\ell}}$ for every $\ell \in [p]$. Thus, $W \subseteq \overline{H_{S-u_p}}$. Because $e \in \delta(W) \setminus \delta(\overline{H_{S-u_p}})$, we have that $e \subseteq W \cup Y_p$, $e \cap W \neq \emptyset$, and $e \cap Y_p \neq \emptyset$. Therefore, in order to show that e has the three desired properties as in the claim, it suffices to show that $Y_p \subseteq Z$. We prove this next.

By definition, $Y_p \cap W = \emptyset$. By Claim 5, for every $i \in [p-1]$, we have that $\overline{H_{S-u_p}} \cap \overline{H_{S-u_i}} \subseteq \overline{H_{S-u_1}}$ and $\overline{H_{S-u_p}} \cap \overline{H_{S-u_i}} \subseteq \overline{H_{S-u_2}}$, so $\overline{H_{S-u_p}} \cap \overline{H_{S-u_i}} \subseteq \overline{H_{S-u_1}} \cap \overline{H_{S-u_2}} \subseteq W$. Thus, for every $i \in [p-1]$, $Y_p \cap Y_i \subseteq \overline{H_{S-u_p}} \cap \overline{H_{S-u_i}} \subseteq W$, so because $Y_p \cap W = \emptyset$, we have that $Y_p \cap Y_i = \emptyset$ for every $i \in [p-1]$. Therefore,

$$Y_p \subseteq \overline{W \cup \left(\bigcup_{i=1}^{p-1} Y_i \right)} = \overline{\bigcup_{i=1}^{p-1} \overline{H_{S-u_i}}} = \bigcap_{i=1}^{p-1} H_{S-u_i} = Z. \quad \square$$

By Claim 6, there is a hyperedge e satisfying the conditions of the second conclusion of Theorem 6. Therefore, by Theorem 6, there exists a k-partition $\mathcal{P}'$ with

$$\begin{aligned} |\delta(\mathcal{P}')| &< \frac{1}{2} \min\{d(A_i) + d(A_j) : i, j \in [p-1], i \neq j\} \\ &= d(U) && \text{(By Claim 3)} \\ &= OPT_{k\text{-cut}}. && \text{(By assumption of the theorem)} \end{aligned}$$

Thus, we have obtained a k-partition $\mathcal{P}'$ with $|\delta(\mathcal{P}')| < OPT_{k\text{-cut}}$, which is a contradiction. This completes the proof of $|S| \leq 2k-1$. □

3. Proof of Theorem 5

We prove Theorem 5 in this section. Applying Theorem 7 to $(\overline{U}, U)$ yields the following corollary.

Corollary 3. *Let $G = (V, E)$ be a hypergraph, and let $OPT_{k\text{-cut}}$ be the optimum value of* Hypergraph-k-Cut *in G for some integer $k \geq 2$. Suppose $(U, \overline{U})$ is a 2-partition of V with $d(U) = OPT_{k\text{-cut}}$. Then, there exists a set $T \subseteq \overline{U}$ with $|T| \leq 2k-1$ such that every minimum (U, T)-terminal cut $(A, \overline{A})$ satisfies $\delta(A) = \delta(U)$.*

We now restate Theorem 5 and prove it using Theorem 7 and Corollary 3.

Theorem 5 (Restated). *Let $G = (V, E)$ be a hypergraph and let $OPT_{k\text{-cut}}$ be the optimum value of* Hypergraph-k-Cut *in G for some integer $k \geq 2$. Suppose $(U, \overline{U})$ is a 2-partition of V with $d(U) = OPT_{k\text{-cut}}$. Then, there exist sets $S \subseteq U$, $T \subseteq \overline{U}$ with $|S| \leq 2k-1$ and $|T| \leq 2k-1$ such that every minimum (S, T)-terminal cut $(A, \overline{A})$ satisfies $\delta(A) = \delta(U)$.*

Proof. By Theorem 7, there exists a subset $S \subseteq U$ with $|S| \leq 2k-1$ such that every minimum $(S, \overline{U})$-terminal cut $(A, \overline{A})$ has $\delta(A) = \delta(U)$. By Corollary 3, there exists a subset $T \subseteq \overline{U}$ with $|T| \leq 2k-1$ such that every minimum (U, T)-terminal cut $(A, \overline{A})$ has $\delta(A) = \delta(U)$. We show that every minimum (S, T)-terminal cut $(A, \overline{A})$ has $\delta(A) = \delta(U)$. We need the following claim.

Claim 7. *Let $(Y, \overline{Y})$ be the source minimal minimum (S, T)-terminal cut. Then, $\delta(Y) = \delta(U)$.*

Proof. Because $(U, \overline{U})$ is a (S, T)-terminal cut, and $(Y, \overline{Y})$ is a minimum (S, T)-terminal cut, we have that

$$d(U) \geq d(Y).$$

Because $(U \cap Y, \overline{U \cap Y})$ is a $(S, \overline{U})$-terminal cut, we have that

$$d(U \cap Y) \geq d(U).$$

Because $(U \cup Y, \overline{U \cup Y})$ is a (U, T)-terminal cut, we have that

$$d(U \cup Y) \geq d(U).$$

Thus, by the submodularity of the hypergraph cut function, we have that

$$2d(U) \geq d(U) + d(Y) \geq d(U \cap Y) + d(U \cup Y) \geq 2d(U).$$

Therefore, we have that $d(U \cap Y) = d(U)$ and $d(Y) = d(U)$. Further, because $(Y, \overline{Y})$ is a minimum (S, T)-terminal cut, it follows that $(U \cap Y, \overline{U \cap Y})$ is a minimum (S, T)-terminal cut. Because $(Y, \overline{Y})$ is the source minimal (S, T)-terminal cut, we have that $U \cap Y = Y$, and hence, $Y \subseteq U$. Therefore, $(Y, \overline{Y})$ is a minimum $(S, \overline{U})$-terminal cut. By the choice of S, we have that $\delta(Y) = \delta(U)$. □

Applying Claim 7 to both sides of the partition $(U, \overline{U})$, we have that the source minimal minimum (S, T)-terminal cut $(Y, \overline{Y})$ has $\delta(Y) = \delta(U)$, and the source minimal minimum (T, S)-terminal cut $(Z, \overline{Z})$ has $\delta(Z) = \delta(U)$. Therefore, for every $e \in \delta(U)$, we have that $e \cap Y \neq \emptyset$ and $e \cap Z \neq \emptyset$.

Let $(A, \overline{A})$ be a minimum (S, T)-terminal cut. Because $(Y, \overline{Y})$ is the source minimal minimum (S, T)-terminal cut, we have that $Y \subseteq A$. Because $(Z, \overline{Z})$ is the source minimal minimum (T, S)-terminal cut, we have that $Z \subseteq \overline{A}$. Because every $e \in \delta(U)$ intersects both Y and Z, it follows that every $e \in \delta(U)$ intersects both A and $\overline{A}$, and hence, $\delta(U) \subseteq \delta(A)$. Because $(A, \overline{A})$ is a minimum (S, T)-terminal cut, $d(A) \leq d(U)$, and thus, we have that $\delta(A) = \delta(U)$. □

4. Algorithm for Enum-Hypergraph-*k*-Cut

In this section, we design a deterministic algorithm for Enum-Hypergraph-*k*-Cut that is based on divide and conquer and has a runtime of $n^{O(k)}$ source minimal minimum (s, t)-terminal cut computations, where n is the number of vertices in the input hypergraph. The high-level idea is to use minimum (S, T)-terminal cuts to enumerate a collection of candidate cuts such that, for every optimum k-partition for Hypergraph-*k*-Cut, either the union of some $k/2$ parts of the optimum k-partition is contained in the candidate collection, or we find the set of hyperedges crossing this optimum k-partition. This helps in cutting the recursion depth to $\log k$, which saves on overall runtime. We describe the algorithm in Figure 2 and its guarantees in Theorem 8. We recall that for a hypergraph $G = (V, E)$ and a subset $A \subseteq V$, the subgraph $G[A]$ induced by A is given by $G[A] = (A, E')$, where $E' := \{e \in E : e \subseteq A\}$.

Theorem 8 is a self-contained proof that the number of min-*k*-cut-sets in a n-vertex hypergraph is $O(n^{8k \log k})$ and the runtime of the algorithm in Figure 2 is $O(n^{8k \log k})$ source minimal minimum (s, t)-terminal cut computations. In Lemma 2, we improve the runtime analysis of the same algorithm to $O(n^{16k})$ source minimal minimum (s, t)-terminal cut computations. For this, we exploit the known fact that the number of min-*k*-cut-sets in a n-vertex hypergraph is $O(n^{2k-2})$ (via the randomized algorithm in Chandrasekaran et al. [16]).

Theorem 8 and Lemma 2 together imply Theorem 2 because the source minimal minimum (s, t)-terminal cut in a n-vertex hypergraph of size p can be computed in time $O(np)$ (Chekuri and Xu [18]).

Theorem 8. *Let $G = (V, E)$ be an n-vertex hypergraph of size p, and let k be a positive integer. Then, algorithm Enum-Cut-Sets(G, k) in Figure 2 returns the family of all* min-*k*-cut-sets *in G, and it can be implemented to run in time at most $D^{4k} n^{(8k-6)\log k} T(n, p)$ for some constant D, where $T(n, p)$ denotes the time complexity for computing the source minimal*

Figure 2. Divide and conquer algorithm to enumerate hypergraph minimum k-cut-sets.

Algorithm Enum-Cut-Sets$(G=(V,E),k)$
Input: Hypergraph $G=(V,E)$ and an integer $k\geq 1$
Output: Family of all MIN-k-CUT-SETs in G
If $k=1$
  Return $\{\emptyset\}$
Else
  Initialize $\mathcal{C}\leftarrow\emptyset$, $\mathcal{F}\leftarrow\emptyset$
  For each pair (S,T) such that $S,T\subseteq V$ with $S\cap T=\emptyset$ and $|S|,|T|\leq 2k-1$
    Compute the source minimal minimum (S,T)-terminal cut $(A,\overline{A})$
    If $G-\delta(A)$ has at least k connected components
      $\mathcal{F}\leftarrow\mathcal{F}\cup\{\delta(A)\}$
    Else
      $\mathcal{C}\leftarrow\mathcal{C}\cup\{A\}$
  For each $A\in\mathcal{C}$ such that $|A|\geq\lfloor k/2\rfloor$ and $|\overline{A}|\geq k-\lfloor k/2\rfloor$
    $\mathcal{F}_A\leftarrow$Enum-Cut-Sets$(G[A],\lfloor k/2\rfloor)$
    $\mathcal{F}'_A\leftarrow$Enum-Cut-Sets$(G[\overline{A}],k-\lfloor k/2\rfloor)$
    $\mathcal{F}\leftarrow\mathcal{F}\cup\{\delta(A)\cup F\cup F':F\in\mathcal{F}_A,F'\in\mathcal{F}'_A\}$
  Among all k-cut-sets in the family $\mathcal{F}$, return the subfamily that are of smallest size

minimum (s,t)-terminal cut in an n-vertex hypergraph of size p. Moreover, the cardinality of the family returned by the algorithm is at most $C^{2k-1}n^{(8k-6)\log k}$ for some constant C.

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

For the base case of $k=1$, the only MIN-k-CUT-SET is the empty set that is contained in the returned family. We now show the induction step. Assume that $k\geq 2$. Let $F\subseteq E$ be a MIN-k-CUT-SET in G and let $(V_1,\ldots,V_k)$ be an optimum k-partition for HYPERGRAPH-k-CUT such that $F=\delta(V_1,\ldots,V_k)$. We show that F is in the family returned by the algorithm. Let $U:=\cup_{i=1}^{\lfloor k/2\rfloor}V_i$. We distinguish between the following two cases:

1. Suppose $d(U)<OPT_{k\text{-cut}}$.

By Theorem 3, there exist disjoint subsets $S,T\subseteq V$ with $|S|,|T|\leq 2k-2$ such that $(U,\overline{U})$ is the unique minimum (S,T)-terminal cut. Hence, the set U is in the collection $\mathcal{C}$. Moreover, U contains $\lfloor k/2\rfloor$ nonempty sets $V_1,V_2,\ldots,V_{\lfloor k/2\rfloor}$, so we have $|U|\geq\lfloor k/2\rfloor$. Similarly, we have $|\overline{U}|\geq k-\lfloor k/2\rfloor$. Because $(V_1,\ldots,V_k)$ is an optimum k-partition for HYPERGRAPH-k-CUT, the set $\{e\in F:e\subseteq U\}$ is a MIN-$\lfloor K/2\rfloor$-CUT-SET in $G[U]$. Similarly, the set $\{e\in F:e\subseteq\overline{U}\}$ is a MIN-$(K-\lfloor K/2\rfloor)$-CUT-SET in $G[\overline{U}]$. Because $d(U)<OPT_{k\text{-cut}}$, we know that $G-\delta(U)$ has fewer than k connected components. Therefore, the set U is in the collection $\mathcal{C}$. By induction hypothesis, we know that the set $\{e\in F:e\subseteq U\}$ is contained in the family $\mathcal{F}_U$ and the set $\{e\in F:e\subseteq\overline{U}\}$ is contained in the family $\mathcal{F}'_U$. Therefore, the set F is added to the family $\mathcal{F}$ in the second for-loop.

2. Suppose $d(U)=OPT_{k\text{-cut}}$.

By Theorem 5, there exist sets $S\subseteq U$ and $T\subseteq\overline{U}$ with $|S|,|T|\leq 2k-1$ such that the source minimal minimum (S,T)-terminal cut $(A,\overline{A})$ satisfies $\delta(A)=\delta(U)=F$. Therefore, the set A is in the collection $\mathcal{C}$. Because $F=\delta(A)$, the hypergraph $G-\delta(A)$ contains at least k connected components. Therefore, the set $F=\delta(A)$ is added to the family $\mathcal{F}$ in the first for-loop.

Thus, in both cases, we show that the set F is contained in the family $\mathcal{F}$. Because the algorithm returns the subfamily of hyperedge sets in $\mathcal{F}$ that are MIN-k-CUT-SETS, the set F is in the family returned by the algorithm.

Next, we bound the cardinality of the family returned by the algorithm. Let $f(k,n)$ be the maximum cardinality of the family returned by the algorithm among all n-vertex hypergraphs. There are $O(n^{4k-2})$ pairs of subsets $S,T\subseteq V$ with $|S|,|T|\leq 2k-1$ and $S\cap T=\emptyset$. Hence, the cardinality of the collection $\mathcal{C}$ and the family $\mathcal{F}$ at the end of the first for-loop is $O(n^{4k-2})$. Consequently, for $k\geq 2$, by the recursion, we have that

$$f(k,n)=O(n^{4k-2})f\left(\left\lfloor\frac{k}{2}\right\rfloor,n\right)f\left(k-\left\lfloor\frac{k}{2}\right\rfloor,n\right) \tag{8}$$

and $f(1,n)=1$. We now show that $f(k,n)\leq C^{2k-1}n^{(8k-6)\log k}$ for some constant C.

Let $C \geq 1$ be a constant larger than the constant occurring in the O-notation of (8). Then, we have that

$$f(k,n) \leq Cn^{4k-2} f\left(\left\lfloor \frac{k}{2} \right\rfloor, n\right) f\left(k - \left\lfloor \frac{k}{2} \right\rfloor, n\right). \tag{9}$$

We show by induction that $f(k,n) \leq C^{2k-1} n^{(8k-6)\log k}$. The base case of $k=1$ holds because $f(1,n)=1$. For the induction step, let $k \geq 2$. Using (9) and the induction hypothesis, we have that

$$\begin{aligned} f(k,n) &\leq Cn^{4k-2} \cdot C^{2\lfloor \frac{k}{2} \rfloor - 1} n^{\left(8\lfloor \frac{k}{2} \rfloor - 6\right)\log\left(\lfloor \frac{k}{2} \rfloor\right)} \cdot C^{2k-2\lfloor \frac{k}{2} \rfloor - 1} n^{\left(8k - 8\lfloor \frac{k}{2} \rfloor - 6\right)\log\left(k - \lfloor \frac{k}{2} \rfloor\right)} \\ &= C^{2k-1} \cdot n^{4k-2+\left(8\lfloor \frac{k}{2} \rfloor - 6\right)\log\left(\lfloor \frac{k}{2} \rfloor\right) + \left(8k - 8\lfloor \frac{k}{2} \rfloor - 6\right)\log\left(k - \lfloor \frac{k}{2} \rfloor\right)}. \end{aligned}$$

Here, the exponent of n is

$$\begin{aligned} &4k - 2 + \left(8\left\lfloor \frac{k}{2} \right\rfloor - 6\right)\log\left(\left\lfloor \frac{k}{2} \right\rfloor\right) + \left(8k - 8\left\lfloor \frac{k}{2} \right\rfloor - 6\right)\log\left(k - \left\lfloor \frac{k}{2} \right\rfloor\right) \\ &\leq 4k - 2 + (8k-12)\log\left(\frac{k+1}{2}\right), \end{aligned}$$

and it suffices to show that

$$4k - 2 + (8k-12)\log\left(\frac{k+1}{2}\right) \leq (8k-6)\log k. \tag{10}$$

This follows immediately for $k=2$. For $k \geq 3$, we have that

$$\begin{aligned} &4k - 2 + (8k-12)\log\left(\frac{k+1}{2}\right) - (8k-6)\log k \\ &= 4k - 2 + (8k-12)\log(k+1) - 8k + 12 - (8k-6)\log k \\ &= (8k-6)\log\frac{k+1}{k} - 6\log(k+1) - 4k + 10 \\ &\leq (8k-6)\log(4/3) - 6\log(k+1) - 4k + 10 \qquad (\text{since } k \geq 3) \\ &\leq \frac{1}{2}(8k-6) - 12 - 4k + 10 \\ &< 0. \end{aligned}$$

This completes the proof that $f(k,n) \leq C^{2k-1} n^{(8k-6)\log k}$.

Next, we bound the maximum runtime of the algorithm. Let $N(k,n)$ denote the runtime of the algorithm among all n-vertex hypergraphs. We have $N(1,n) = O(1)$. For $k \geq 2$, there are $O(n^{4k-2})$ pairs of subsets $S, T \subseteq V$ with $|S|, |T| \leq 2k-1$ and $S \cap T = \emptyset$. Hence, the first for-loop performs $O(n^{4k-2})$ source minimal minimum (S,T)-terminal cut computations. The collection $\mathcal{C}$ and the family $\mathcal{F}$ at the end of the first for-loop each have $O(n^{4k-2})$ sets. This implies that the first for-loop can be implemented to run in $O(n^{4k-2})T(n,p)$ time. For each $A \in \mathcal{C}$, the computation of Enum-Cut-Sets$(G[A], \lfloor k/2 \rfloor)$ in the second for-loop runs in $N(\lfloor k/2 \rfloor, n)$ time. The computation of Enum-Cut-Sets$(G[\overline{A}], k - \lfloor k/2 \rfloor)$ in the second for-loop runs in $N(k - \lfloor k/2 \rfloor, n)$ time. Hence, the second for-loop can be implemented to run in

$$O(n^{4k-2})\left(N\left(\left\lfloor \frac{k}{2} \right\rfloor, n\right) + N\left(k - \left\lfloor \frac{k}{2} \right\rfloor, n\right) + f\left(\left\lfloor \frac{k}{2} \right\rfloor, n\right) \cdot f\left(k - \left\lfloor \frac{k}{2} \right\rfloor, n\right) \cdot O(p)\right)$$

time. The last step to prune the family $\mathcal{F}$ can be implemented to run in time that is linear in the time to implement the first and second for-loops: this is because, for each member of $\mathcal{F}$, we can decide whether it is a minimum k-cut set in time linear in the time to write this member in $\mathcal{F}$. Therefore, we have

$$\begin{aligned} N(k,n) &= O(n^{4k-2})T(n,p) \\ &\quad + O(n^{4k-2})\left(N\left(\left\lfloor \frac{k}{2} \right\rfloor, n\right) + N\left(k - \left\lfloor \frac{k}{2} \right\rfloor, n\right) + f\left(\left\lfloor \frac{k}{2} \right\rfloor, n\right) \cdot f\left(k - \left\lfloor \frac{k}{2} \right\rfloor, n\right) \cdot O(p)\right). \end{aligned} \tag{11}$$

We now show that $N(k,n) \leq D^{4k} n^{(8k-6)\log k} T(n,p)$ for some constant D.

Let $D \geq \max\{C, N(1,n), 4\}$ be a constant that is larger than the constants occurring in the O-notation of (11). Then, we have that

$$N(k,n) \leq Dn^{4k-2}\left(T(n,p) + N\left(\left\lfloor \frac{k}{2}\right\rfloor, n\right) + N\left(k - \left\lfloor \frac{k}{2}\right\rfloor, n\right) + f\left(\left\lfloor \frac{k}{2}\right\rfloor, n\right)\cdot f\left(k - \left\lfloor \frac{k}{2}\right\rfloor, n\right)\cdot Dp\right). \tag{12}$$

We show by induction that $N(k,n) \leq D^{4k} n^{(8k-6)\log k} T(n,p)$. The base case of $k=1$ holds because $N(1,n) \leq D$. We now show the induction step. Let $k \geq 2$. We use the fact that $T(n,p) \geq p$. Then, using (12), induction hypothesis, and the bound $f(k,n) \leq C^{2k-1} n^{(8k-6)\log k}$ shown earlier, we have that

$$\begin{aligned}
N(k,n) &\leq Dn^{4k-2}\left(T(n,p) + N\left(\left\lfloor \frac{k}{2}\right\rfloor, n\right) + N\left(k - \left\lfloor \frac{k}{2}\right\rfloor, n\right)\right)\\
&\quad + Dn^{4k-2}\left(C^{2k-2} n^{\left(8\lfloor \frac{k}{2}\rfloor - 6\right)\log\left(\lfloor \frac{k}{2}\rfloor\right)} n^{\left(8k - 8\lfloor \frac{k}{2}\rfloor - 6\right)\log\left(k - \lfloor \frac{k}{2}\rfloor\right)}\cdot Dp\right)\\
&\leq Dn^{4k-2}\left(T(n,p) + N\left(\left\lfloor \frac{k}{2}\right\rfloor, n\right) + N\left(k - \left\lfloor \frac{k}{2}\right\rfloor, n\right) + C^{2k-2} n^{(8k-12)\log\left(\frac{k+1}{2}\right)}\cdot Dp\right)\\
&\leq Dn^{4k-2}\left(T(n,p) + D^{4\lfloor \frac{k}{2}\rfloor} n^{\left(8\lfloor \frac{k}{2}\rfloor - 6\right)\log\left(\lfloor \frac{k}{2}\rfloor\right)} T(n,p)\right)\\
&\quad + Dn^{4k-2}\left(D^{4k-4\lfloor \frac{k}{2}\rfloor} n^{\left(8k - 8\lfloor \frac{k}{2}\rfloor - 6\right)\log\left(k - \lfloor \frac{k}{2}\rfloor\right)} T(n,p) + C^{2k-2} n^{(8k-12)\log\left(\frac{k+1}{2}\right)}\cdot Dp\right)\\
&\leq Dn^{4k-2}\left(T(n,p) + 2D^{2k+2} n^{(4k-2)\log\left(\frac{k+1}{2}\right)} T(n,p) + C^{2k-2} n^{(8k-12)\log\left(\frac{k+1}{2}\right)}\cdot DT(n,p)\right) \qquad (13)\\
&= T(n,p)\left(Dn^{4k-2} + 2D^{2k+3} n^{(4k-2)\log(k+1)} + C^{2k-2} D^2 n^{(8k-12)\log\left(\frac{k+1}{2}\right)+4k-2}\right)\\
&\leq T(n,p)\left(Dn^{4k-2} + 2D^{2k+3} n^{(4k-2)\log(k+1)} + C^{2k-2} D^2 n^{(8k-6)\log k}\right) \qquad \text{(by inequality (10))}\\
&\leq T(n,p)\cdot n^{(8k-6)\log k}(D + 2D^{2k+3} + C^{2k-2} D^2) \qquad (14)\\
&\leq T(n,p)\cdot n^{(8k-6)\log k}\cdot 4\max(C,D)^{2k+3}\\
&\leq D^{4k} n^{(8k-6)\log k} T(n,p),
\end{aligned}$$

where Inequality (13) is because $\lfloor k/2\rfloor, k - \lfloor k/2\rfloor \leq (k+1)/2$ and Inequality (14) is because $(4k-2)\log(k+1) \leq (8k-6)\log k$ for all $k \geq 2$. □

We recall that the number of MIN-k-CUT-SETS in an n-vertex hypergraph is $O(n^{2k-2})$ (Chandrasekaran et al. [16]). Assuming this bound improves the runtime of algorithm Enum-Cut-Sets(G, k) in Figure 2.

Lemma 2. *Algorithm Enum-Cut-Sets(G, k) in Figure 2 can be implemented to run in time at most $C^{2k} n^{16k-26} T(n,p)$ for some constant C, where n is the number of vertices, p is the size of the input hypergraph G, and T(n, p) denotes the time complexity for computing the source minimal minimum (s, t)-terminal cut in an n-vertex hypergraph of size p.*

Proof. Let $N(k, n)$ denote the maximum runtime of the algorithm among all n-vertex hypergraphs. Then, we have that

$$N(1,n) = O(1) \tag{15}$$

because the algorithm runs in constant time. For $k \geq 2$, there are $O(n^{4k-2})$ pairs of subsets $S, T \subseteq V$ with $|S|, |T| \leq 2k-1$ and $S \cap T = \emptyset$. Hence, the first for-loop performs $O(n^{4k-2})$ source minimal minimum (S, T)-terminal cut computations. The collection $\mathcal{C}$ and the family $\mathcal{F}$ at the end of the first for-loop each have $O(n^{4k-2})$ sets. This implies that the first for-loop can be implemented to run in $O(n^{4k-2})T(n,p)$ time. For each $A \in \mathcal{C}$, the computation of Enum-Cut-Sets($G[A], \lfloor k/2\rfloor$) in the second for-loop runs in $N(\lfloor k/2\rfloor, n)$ time. The computation of Enum-Cut-Sets($G[\overline{A}], k - \lfloor k/2\rfloor$) in the second for-loop runs in $N(k - \lfloor k/2\rfloor, n)$ time. We recall that $\mathcal{F}_A$ consists of all minimum $\lfloor k/2\rfloor$-cut-sets in a n-vertex graph and, hence, has size $O(n^{2\lfloor k/2\rfloor - 2})$. Similarly, $\mathcal{F}'_A$ has size $O(n^{2(k - \lfloor k/2\rfloor) - 2})$. Hence, the second

for-loop can be implemented to run in time

$$O(n^{4k-2})\left(N\left(\left\lfloor\frac{k}{2}\right\rfloor,n\right)+N\left(k-\left\lfloor\frac{k}{2}\right\rfloor,n\right)+O(n^{2\lfloor k/2\rfloor-2})O(n^{2(k-\lfloor k/2\rfloor)-2})\right)$$
$$=O(n^{4k-2})\left(N\left(\left\lfloor\frac{k}{2}\right\rfloor,n\right)+N\left(k-\left\lfloor\frac{k}{2}\right\rfloor,n\right)+O(n^{2k-4})\right).$$

Moreover, the size of the family $\mathcal{F}$ at the end of the second for-loop is $O(n^{4k-2})+O(n^{4k-2})|\mathcal{F}_A|\cdot|\mathcal{F}'_A|=O(n^{4k-2})$ $O(n^{2k-4})=O(n^{6k-6})$. Hence, the last step to prune the family $\mathcal{F}$ can be implemented to run in time $O(n^{6k-6})$. Hence, the second for-loop and the last step can together be implemented to run in time

$$O(n^{4k-2})\left(N\left(\left\lfloor\frac{k}{2}\right\rfloor,n\right)+N\left(k-\left\lfloor\frac{k}{2}\right\rfloor,n\right)+O(n^{2k-4})\right).$$

Therefore, we have

$$N(k,n)=O(n^{4k-2})\left(T(n,p)+N\left(\left\lfloor\frac{k}{2}\right\rfloor,n\right)+N\left(k-\left\lfloor\frac{k}{2}\right\rfloor,n\right)+O(n^{2k-4})\right). \tag{16}$$

We now solve the recursion in (16) under the base case given by (15) to show that $N(k,n)=O(n^{\max(0,16k-26)})$ $T(n,p)$.

Let $C\geq 4$ be a constant that is larger than the constants that appear in the O-notations of (15) and (16). Then, we have that $N(1,n)\leq C$ and

$$N(k,n)\leq Cn^{4k-2}\left(T(n,p)+N\left(\left\lfloor\frac{k}{2}\right\rfloor,n\right)+N\left(k-\left\lfloor\frac{k}{2}\right\rfloor,n\right)+Cn^{2k-4}\right). \tag{17}$$

We show that $N(k,n)\leq C^{2k}n^{\max(0,16k-26)}T(n,p)$ by induction on k. For the base case of $k=1$, we have that $N(1,n)$ $\leq C$. We now show the induction step.

For $k=2$, using Inequality (17) and $N(1,n)\leq C$, we have

$$N(2,n)\leq Cn^6\left(T(n,p)+2N(1,n)+C\right)\leq Cn^6\left(T(n,p)+2C+C\right)$$
$$\leq Cn^6\cdot 4CT(n,p)\leq C^4n^6T(n,p)=C^{2k}n^{\max(0,16k-26)}T(n,p).$$

For $k=3$, using Inequality (17), $N(1,n)\leq C$, and $N(2,n)\leq C^4n^6T(n,p)$, we have

$$N(3,n)\leq Cn^{10}\left(T(n,p)+N(1,n)+N(2,n)+Cn^2\right)\leq Cn^{10}\left(T(n,p)+C+C^4n^6T(n,p)+Cn^2\right)$$
$$\leq Cn^{10}\cdot 4C^4n^6T(n,p)\leq C^6n^{22}T(n,p)=C^{2k}n^{\max(0,16k-26)}T(n,p).$$

Let $k\geq 4$. We have that $16\lfloor k/2\rfloor-26\geq 0$ and $16(k-\lfloor k/2\rfloor)-26\geq 0$. Using (17) and the induction hypothesis, we have that

$$N(k,n)\leq Cn^{4k-2}\left(T(n,p)+C^{2\lfloor\frac{k}{2}\rfloor}n^{16\lfloor\frac{k}{2}\rfloor-26}T(n,p)+C^{2\left(k-\lfloor\frac{k}{2}\rfloor\right)}n^{16\left(k-\lfloor\frac{k}{2}\rfloor\right)-26}T(n,p)+Cn^{2k-4}\right)$$
$$\leq Cn^{4k-2}\left(1+C^{2\lfloor\frac{k}{2}\rfloor}n^{16\lfloor\frac{k}{2}\rfloor-26}+C^{2\left(k-\lfloor\frac{k}{2}\rfloor\right)}n^{16\left(k-\lfloor\frac{k}{2}\rfloor\right)-26}+Cn^{2k-4}\right)T(n,p). \tag{18}$$

Suppose k is even. Then, using (18), we have that

$$N(k,n)\leq Cn^{4k-2}(1+2C^kn^{8k-26}+Cn^{2k-4})T(n,p)$$
$$\leq C^{k+1}n^{4k-2}(1+2n^{8k-26}+n^{2k-4})T(n,p)$$
$$\leq C^{k+1}n^{4k-2}(4n^{\max\{0,8k-26,2k-4\}})T(n,p)$$
$$\leq C^{2k}n^{16k-26}T(n,p),$$

where the last inequality is because $C^{2k} \geq 4C^{k+1}$ and $\max\{0, 8k-26, 2k-4\} \leq 12k-24$ for $k \geq 2$. Next, suppose that k is odd. Then, using (18), we have that

$$\begin{aligned} N(k,n) &\leq Cn^{4k-2}\left(1 + C^{k-1}n^{8(k-1)-26} + C^{k+1}n^{8(k+1)-26} + Cn^{2k-4}\right)T(n,p) \\ &\leq C^{k+2}n^{4k-2}\left(1 + n^{8(k-1)-26} + n^{8(k+1)-26} + n^{2k-4}\right)T(n,p) \\ &\leq C^{k+2}n^{4k-2}(4n^{\max\{0,8k-18,2k-4\}})T(n,p) \\ &\leq C^{2k}n^{16k-26}T(n,p), \end{aligned}$$

where the last inequality is because $C^{2k} \geq 4C^{k+2}$ and $\max\{0, 8k-18, 2k-4\} \leq 12k-24$ for $k \geq 4$. This completes the proof that $N(k,n) \leq C^{2k}n^{16k-26}T(n,p)$. □

5. Algorithm for Enum-MinMax-Hypergraph-*k*-Partition

In this section, we design a deterministic algorithm for Enum-MinMax-Hypergraph-*k*-Partition that runs in time $n^{O(k^2)}p$, where n is the number of vertices and p is the size of the input hypergraph. For this, we rely on the notion of k-cut-set representatives.

We recall that, for a k-partition $(V_1, \ldots, V_k)$ and disjoint subsets $U_1, \ldots, U_k \subseteq V$, the k-tuple $(U_1, \ldots, U_k)$ is defined to be a k-cut set representative of $(V_1, \ldots, V_k)$ if $U_i \subseteq V_i$ and $\delta(U_i) = \delta(V_i)$ for all $i \in [k]$. We first show that there exists a polynomial-time algorithm to verify whether a given k-tuple $(U_1, \ldots, U_k)$ is a k-cut set representative.

Theorem 9. *Let $G = (V, E)$ be an n-vertex hypergraph of size p and let k be a positive integer. Then, there exists an algorithm, denoted algorithm Recover-Partition, that takes as input the hypergraph G and disjoint subsets $U_1, \ldots, U_k \subseteq V$ and runs in time $O(knp)$ to decide if $(U_1, \ldots, U_k)$ is a k-cut set representative of some k-partition $(V_1, \ldots, V_k)$ and, if so, then return such a k-partition.*

Proof. We use algorithm Recover-Partition$(G, U_1, \ldots, U_k)$ in Figure 3.

We begin by showing correctness. Because Algorithm 3 maintains $U_i \subseteq P_i$ and $\delta(U_i) = \delta(P_i)$ for all $i \in [k]$, if it returns a k-partition, then the k-partition necessarily satisfies the required conditions. Next, we show that, if $(U_1, \ldots, U_k)$ is a k-cut set representative of a k-partition $(V_1, \ldots, V_k)$, then the algorithm indeed returns a k-partition $(P_1, \ldots, P_k)$ with $U_i \subseteq P_i$ and $\delta(U_i) = \delta(P_i)$ for all $i \in [k]$ (however, $(P_1, \ldots, P_k)$ may not necessarily be the same as $(V_1, \ldots, V_k)$).

Let $(V_1, \ldots, V_k)$ be a k-partition such that $U_i \subseteq V_i$ and $\delta(U_i) = \delta(V_i)$ for all $i \in [k]$. Let $(P_1, \ldots, P_k)$ be the sequence of subsets at the end of the for-loop. Moreover, for each $j \in [t]$, let $(P_1^j, \ldots, P_k^j)$ be the sequence of subsets at the end of the jth iteration of the for-loop. For notational convenience, for $i \in [k]$, we define $P_i^0 := U_i$. We note that $(P_1, \ldots, P_k) = (P_1^t, \ldots, P_k^t)$ and $P_i^0 \subseteq P_i^1 \subseteq \cdots \subseteq P_i^t$ for every $i \in [k]$. We observe that $U_i \subseteq P_i^j$ and $\delta(U_i) = \delta(P_i^j)$ for all $j \in \{0, 1, 2, \ldots, t\}$. Moreover, the subsets $P_1^j, \ldots, P_k^j$ are pairwise disjoint for each $j \in \{0, 1, 2, \ldots, t\}$. Therefore, it suffices to show that $\cup_{i=1}^k P_i = V$.

We claim that $C_1 \cup \cdots \cup C_j \subseteq \cup_{i=1}^k P_i^j$ for each $j \in \{0, 1, \ldots, t\}$. Applying this claim for $j = t$ gives that $\cup_{i=1}^k P_i = V$ as desired. We now show the claim by induction on j. The base case of $j = 0$ holds by definition. We now prove the induction step. By induction hypothesis, we have that $C_1 \cup \ldots \cup C_{j-1} \subseteq \cup_{i=1}^k P_i^{j-1}$. We show that there exists $i \in [k]$

Figure 3. Algorithm for Theorem 9.

Algorithm Recover-Partition$(G = (V, E), U_1, \ldots, U_k)$
Input: Hypergraph $G = (V, E)$ and disjoint subsets $U_1, \ldots, U_k \subseteq V$
Output: Decide if there exists a k-partition $(V_1, \ldots, V_k)$ of V with $U_i \subseteq V_i$ and $\delta(U_i) = \delta(V_i)$ $\forall i \in [k]$, and return one if it exists
Initialize $P_i \leftarrow U_i$ for all $i \in [k]$
Let $C_1, \ldots, C_t \subseteq V$ be the components of $G - \cup_{i=1}^k \delta(U_i)$ that are disjoint from $\cup_{i=1}^k U_i$
For $j = 1, \ldots, t$
    If $\exists i \in [k]$ such that $\delta(P_i \cup C_j) = \delta(P_i)$
        $P_i \leftarrow P_i \cup C_j$
If $(P_1, \ldots, P_k)$ is a k-partition of V
    Return $(P_1, \ldots, P_k)$
Else
    Return NO

such that $\delta(P_i^{j-1} \cup C_j) = \delta(P_i^{j-1})$. We know that C_j is contained in one of the sets in $\{V_1, \dots, V_k\}$, say $C_j \subseteq V_\ell$ for some $\ell \in [k]$. We prove that $\delta(P_\ell^{j-1} \cup C_j) = \delta(P_\ell^j)$ to complete the proof of the claim. Because C_j is a component of $G - \cup_{i=1}^k \delta(U_i) = G - \cup_{i=1}^k \delta(V_i)$, we know that each hyperedge in $\delta(C_j)$ crosses the k-partition $(V_1, \dots, V_k)$. Moreover, each hyperedge in $\delta(C_j)$ intersects $C_j \subseteq V_\ell$, and hence, $\delta(C_j) \subseteq \delta(V_\ell)$. Therefore,

$$\delta(P_\ell^{j-1} \cup C_j) - \delta(P_\ell^{j-1}) \subseteq \delta(C_j) \subseteq \delta(V_\ell) = \delta(U_\ell) = \delta(P_\ell^{j-1}).$$

This is possible only if $\delta(P_\ell^{j-1} \cup C_j) - \delta(P_\ell^{j-1}) = \emptyset$, that is, $\delta(P_\ell^{j-1} \cup C_j) \subseteq \delta(P_\ell^{j-1})$. We now show the reverse inclusion. We have that

$$\begin{aligned}
\delta(P_\ell^{j-1}) - \delta(P_\ell^{j-1} \cup C_j) &= \delta(U_\ell) - \delta(P_\ell^{j-1} \cup C_j) \\
&= \delta(V_\ell) - \delta(P_\ell^{j-1} \cup C_j) \\
&= E(V_\ell - P_\ell^{j-1} - C_j, V - V_\ell - P_\ell^{j-1}) \cup E(V_\ell \cap P_\ell^{j-1}, P_\ell^{j-1} - V_\ell) \\
&\subseteq E[V - P_\ell^{j-1}] \cup E[P_\ell^{j-1}].
\end{aligned}$$

We note that the left-hand side is a subset of $\delta(P_\ell^{j-1})$, whereas the right-hand side is disjoint from $\delta(P_\ell^{j-1})$ because $E[V - P_\ell^{j-1}] \cap \delta(P_\ell^{j-1}) = \emptyset$ and $E[P_\ell^{j-1}] \cap \delta(P_\ell^{j-1}) = \emptyset$. Hence, the containment is possible only if $\delta(P_\ell^{j-1}) - \delta(P_\ell^{j-1} \cup C_j) = \emptyset$, and hence, $\delta(P_\ell^{j-1}) \subseteq \delta(P_\ell^{j-1} \cup C_j)$. Consequently, $\delta(P_\ell^{j-1} \cup C_j) = \delta(P_\ell^{j-1})$.

We now bound the runtime. We can compute $\delta(U_i)$ for all $i \in [k]$ in $O(kp)$ time. For every choice of $j \in [t]$, we can verify if there exists $i \in [k]$ such that $\delta(P_i \cup C_j) = \delta(P_i)$ in time $O(kp)$ as follows: For each $i \in [k]$, we can compute $\delta(P_i \cup C_j)$ in $O(p)$ time and verify if $\delta(P_i \cup C_j) = \delta(U_i) = \delta(P_i)$ in $O(p)$ time. The number of iterations of the for-loop is $t \le n$. Hence, the total runtime is $O(knp)$. □

Next, we address the problem of enumerating all MINMAX-k-CUT-SETS. For this, we define a subproblem: namely, ENUM-MINMAX-HYPERGRAPH-k-CUT-SET-REPS. The input here is a hypergraph $G = (V, E)$ and a fixed positive integer k (e.g., $k = 2, 3, 4, \dots$). The goal is to enumerate a family $\mathcal{F}$ of k-cut set representatives satisfying the following two properties:

1. Every k-tuple $(U_1, \dots, U_k)$ in the family $\mathcal{F}$ is a k-cut set representative of some optimum k-partition $(V_1, \dots, V_k)$ for MINMAX-HYPERGRAPH-k-PARTITION.

2. For every optimum k-partition $(V_1, \dots, V_k)$ for MINMAX-HYPERGRAPH-k-PARTITION, the family $\mathcal{F}$ contains a k-cut set representative $(U_1, \dots, U_k)$ of $(V_1, \dots, V_k)$.

We note that, if a family $\mathcal{F}$ is a solution to ENUM-MINMAX-HYPERGRAPH-k-CUT-SET-REPS, then returning $\{\cup_{i=1}^k \delta(U_i) : (U_1, \dots, U_k) \in \mathcal{F}\}$ solves ENUM-MINMAX-HYPERGRAPH-k-PARTITION. Hence, it suffices to solve ENUM-MINMAX-HYPERGRAPH-k-CUT-SET-REPS in order to solve ENUM-MINMAX-HYPERGRAPH-k-PARTITION. We describe our algorithm for ENUM-MINMAX-HYPERGRAPH-k-CUT-SET-REPS in Figure 4 and its guarantees in Theorem 10. Theorem 1 follows from Theorem 10.

Figure 4. Algorithm for Theorem 10.

Algorithm Enum-MinMax-Reps($G = (V, E), k$)
Input: Hypergraph $G = (V, E)$ and an integer $k \ge 2$
Output: Family $\mathcal{F}$ of k-cut-set representatives of all optimum k-partitions for MINMAX-HYPERGRAPH-k-PARTITION
Initialize $\mathcal{C} \leftarrow \emptyset$, $\mathcal{D} \leftarrow \emptyset$, and $\mathcal{F} \leftarrow \emptyset$
For each pair (S, T) such that $S, T \subseteq V$ with $S \cap T = \emptyset$ and $|S|, |T| \le 2k - 1$
  Compute the source minimal minimum (S, T)-terminal cut $(U, \overline{U})$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{U\}$
For all $(U_1, \dots, U_k) \in \mathcal{C}^k$ such that $U_1, \dots, U_k$ are pairwise disjoint
  If Recover-Partition($G, U_1, \dots, U_k$) returns a k-partition
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(U_1, \dots, U_k)\}$
$\lambda \leftarrow \min\{\max_{i \in [k]} d(U_i) : (U_1, \dots, U_k) \in \mathcal{D}\}$
For all $(U_1, \dots, U_k) \in \mathcal{D}$ such that $\max_{i \in [k]} d(U_i) = \lambda$:
  $\mathcal{F} \leftarrow \mathcal{F} \cup \{(U_1, \dots, U_k)\}$
Return $\mathcal{F}$

Theorem 10. *Let $G = (V, E)$ be an n-vertex hypergraph of size p and let k be a positive integer. Then, algorithm Enum-MinMax-Reps(G, k) in Figure 4 solves Enum-MinMax-Hypergraph-k-Cut-Set-Reps, and it can be implemented to run in time $O(kn^{4k^2-2k+1}p)$. Moreover, the cardinality of the family returned by the algorithm is $O(n^{4k^2-2k})$.*

Proof. We begin by showing correctness; that is, the family $\mathcal{F}$ returned by the algorithm satisfies properties (1) and (2) mentioned in the definition of Enum-MinMax-Hypergraph-k-Cut-Set-Reps. By the second for-loop, each k-tuple added to the collection $\mathcal{D}$ is a k-cut set representative of some k-partition (it need not necessarily be a k-cut set representative of an optimum k-partition for Minmax-Hypergraph-k-Partition). The algorithm returns a subfamily of $\mathcal{D}$, and hence, it returns a subfamily of k-cut set representatives. We only have to show that a k-cut set representative of an arbitrary optimum k-partition for Minmax-Hypergraph-k-Partition is present in the family $\mathcal{D}$; this guarantees that the value λ computed by the algorithm is exactly $OPT_{\text{minmax-}k\text{-partition}}$, and owing to the way in which the algorithm constructs the family $\mathcal{F}$ from the family $\mathcal{D}$, it follows that the family $\mathcal{F}$ satisfies properties (1) and (2).

Let $OPT_{\text{minmax-}k\text{-partition}}$ denote the optimum value of a minmax k-partition in G, and let $OPT_{k\text{-cut}}$ denote the optimum value of a minimum k-cut in G. We note that $OPT_{\text{minmax-}k\text{-partition}} \le OPT_{k\text{-cut}}$. This is because, if $(P_1, \ldots, P_k)$ is a k-partition with minimum $|\delta(P_1, \ldots, P_k)|$ (i.e., an optimum k-partition for Hypergraph-k-Cut), then

$$OPT_{\text{minmax-}k\text{-partition}} \le \max_{i\in[k]} |\delta(P_i)| \le |\delta(P_1, \ldots, P_k)| = OPT_{k\text{-cut}}.$$

Let $(V_1, \ldots, V_k)$ be an arbitrary optimum k-partition for Minmax-Hypergraph-k-Partition. We show that the family $\mathcal{F}$ returned by the algorithm contains a k-cut set representative of $(V_1, \ldots, V_k)$. We have that $d(V_i) \le OPT_{\text{minmax-}k\text{-partition}} \le OPT_{k\text{-cut}}$ for all $i \in [k]$. Hence, by Theorems 3 and 5, there exist subsets $S_i \subseteq V_i$, $T_i \subseteq V - V_i$ with $|S_i|, |T_i| \le 2k-1$ such that the source minimal minimum (S_i, T_i)-terminal cut $(U_i, \overline{U_i})$ satisfies $\delta(U_i) = \delta(V_i)$ for all $i \in [k]$. Source minimality of the cut $(U_i, \overline{U_i})$ also guarantees that $U_i \subseteq V_i$ for all $i \in [k]$. Hence, the k-tuple $(U_1, \ldots, U_k)$ is a k-cut set representative of $(V_1, \ldots, V_k)$. It remains to show that this k-tuple is indeed present in the families $\mathcal{D}$ and $\mathcal{F}$. We note that the sets $U_1, \ldots, U_k$ are added to the collection $\mathcal{C}$ in the first for-loop. Because the k-tuple $(U_1, \ldots, U_k)$ is a k-cut set representative of the k-partition $(V_1, \ldots, V_k)$, the k-tuple $(U_1, \ldots, U_k)$ is added to the family $\mathcal{D}$ in the second for-loop (by correctness of algorithm Recover-Partition as proved in Theorem 9). Because the family $\mathcal{D}$ contains only k-cut set representatives of k-partitions, it follows that $\lambda = OPT_{\text{minmax-}k\text{-partition}}$ and $(U_1, \ldots, U_k)$ are added to the family $\mathcal{F}$ in the third for-loop. Hence, the k-cut set representative $(U_1, \ldots, U_k)$ of the optimum k-partition $(V_1, \ldots, V_k)$ for Minmax-Hypergraph-k-Partition is present in the family $\mathcal{F}$ returned by the algorithm.

The bound on the size of the family $\mathcal{F}$ returned by the algorithm is

$$|\mathcal{F}| \le |\mathcal{D}| \le |\mathcal{C}|^k = O(n^{k(4k-2)}).$$

Next, we bound the runtime of the algorithm. The first for-loop can be implemented to run in time $O(n^{4k-2})$ $T(n,p)$. The second for-loop executes the algorithm from Theorem 9 $O(n^{4k^2-2k})$ times, and hence, the second for-loop can be implemented to run in time $O(kn^{4k^2-2k+1}p)$. The computation of λ and the third for-loop can be implemented to run in time $O(|\mathcal{D}|) = O(n^{4k^2-2k})$. The time to compute λ is $O(kp)|\mathcal{D}| = O(kn^{4k^2-2k}p)$ because, for each $(U_1, \ldots, U_k) \in \mathcal{D}$, we can compute $\max_{i\in[k]} d(U_i)$ in time $O(kp)$. The third for-loop can also be implemented to run in time $O(kp)|\mathcal{D}| = O(kn^{4k^2-2k}p)$. Hence, the total runtime is $O(n^{4k-2})T(n,p) + O(kn^{4k^2-2k+1}p)$. We recall that $T(n,p) = O(np)$, and hence, the total runtime is $O(kn^{4k^2-2k+1}p)$. □

6. A Lower Bound on the Number of Minmax-k-Cut-Sets

In this section, we show that there exist n-vertex connected graphs for which the number of minmax-k-cut-sets is $n^{\Omega(k^2)}$. In particular, we show the following result.

Lemma 3. *For every positive integer $k \ge 2$, there exists a positive integer n such that the number of optimum k-partitions for Minmax-Graph-k-Partition in the n-vertex complete graph is $n^{\Omega(k^2)}$.*

Proof. Let $k \ge 2$ be fixed, and let $G = (V, E)$ be the complete graph on $n = k(k-1)$ vertices (with all edge weights being uniformly one). We show that $OPT_{\text{minmax-}k\text{-partition}} = (k-1)^3$ and every partition of V into k parts of equal size is an optimum k-partition for Minmax-Graph-k-Partition. Because the number of partitions of V into k parts of equal size is $\Omega(k^n) = \Omega(n^{k^2/2})$, the lemma follows.

First, we show that $OPT_{\text{minmax-}k\text{-partition}} \ge (k-1)^3$. For every partition of V into k nonempty parts, the largest part has at least $k-1$ vertices by the pigeonhole principle and at most $k(k-1) - (k-1) = (k-1)^2$ vertices because each of the remaining $k-1$ parts contain at least one vertex. Therefore, the cut value of the largest part

is at least

$$\min_{x\in\{k-1,\ldots,(k-1)^2\}} x(n-x) = (k-1)^3.$$

The equality follows because $n = k(k-1)$ and the function $f(x) = x(n-x)$ is convex and is minimized at the boundaries. This implies that $OPT_{\text{minmax-}k\text{-partition}} \geq (k-1)^3$.

Next, we show that $OPT_{\text{minmax-}k\text{-partition}} \leq (k-1)^3$ and every partition of V into k parts of equal size is an optimum k-partition for Minmax-Graph-k-Partition. Let $(V_1,\ldots,V_k)$ be an arbitrary k-partition of V such that $|V_i| = k-1$ for all $i \in [k]$. The Minmax-Graph-k-Partition objective value of this k-partition is $(k-1)(n-(k-1)) = (k-1)^3$. Thus, $(V_1,\ldots,V_k)$ is an optimum k-partition for Minmax-Graph-k-Partition in G. □

We note that our example exhibiting $n^{\Omega(k^2)}$ optimum k-partitions for Minmax-Graph-k-Partition has the number of vertices n upper bounded by a function of k. We are not aware of examples that exhibit $n^{\Omega(k^2)}$ optimum k-partitions for Minmax-Graph-k-Partition for fixed k but arbitrary n (e.g., $k = 2,3,4,\ldots$ but n is arbitrary).

7. Conclusion

We show the first polynomial bound on the number of minmax-k-cut-sets in hypergraphs for every fixed k and give a polynomial-time algorithm to enumerate all minmax-k-cut-sets as well as all min-k-cut-sets in hypergraphs for every fixed k. Our main contribution is a structural theorem that is the backbone of the correctness analysis of our enumeration algorithms. In order to enumerate minmax-k-cut-sets in hypergraphs, we introduce the notion of k-cut set representatives and enumerated k-cut set representatives of all optimum k-partitions for Minmax-Hypergraph-k-Partition. Our technique builds on known structural results for Hypergraph-k-Cut and Minmax-Hypergraph-k-Partition (Beideman et al. [7], Chandrasekaran and Chekuri [12, 13]).

The technique underlying our enumeration algorithms is not necessarily novel; we simply rely on minimum (s,t)-terminal cuts. Using fixed-terminal cuts to address global partitioning problems is not a novel technique by itself; it is common knowledge that minimum (s, t)-terminal cuts can be used to solve global minimum cut. However, there are several problems in which naive extensions of this technique fail to lead to efficient algorithms: for example, Graph-k-Cut for an n-vertex graph can be reduced to $O(n^k)$ multiway cut instances, but multiway cut for k terminals is NP-hard even for fixed constant k. Adapting this technique for specific partitioning problems requires careful identification of structural properties. In fact, beautiful structural properties are shown for a rich variety of partitioning problems in combinatorial optimization in order to exploit this technique: for example, it is used (1) to design the first efficient algorithm for Graph-k-Cut (Goldschmidt and Hochbaum [27]), (2) to solve certain constrained submodular minimization problems (Goemans and Ramakrishnan [26], Nägele et al. [51]), and (3) more recently to design fast algorithms for global minimum cut in graphs and for Gomory–Hu trees in unweighted graphs (Abboud et al. [1], Li and Panigrahi [46]). Our use of this technique also relies on identifying and proving a suitable structural property, namely, Theorem 5. The advantage of our structural property is that it simultaneously enables enumeration of min-k-cut-sets as well as minmax-k-cut-sets in hypergraphs, which was not possible via structural theorems that were developed before. Furthermore, it helps in showing the first polynomial bound on the number of minmax-k-cut-sets in hypergraphs for every fixed k. In work subsequent to the present paper, we also show that minimum (s, t)-terminal cuts can be used to enumerate all constant approximate minimum cuts in a given connected graph in polynomial time (Beideman et al. [8]).

We also emphasize a limitation of our technique. Although it helps in solving Enum-Hypergraph-k-Cut and Enum-MinMax-Hypergraph-k-Partition, it does not help in solving a seemingly related hypergraph k-partitioning problem: namely, given a hypergraph $G = (V,E)$ and a fixed integer k, find a k-partition $(V_1,\ldots,V_k)$ of the vertex set that minimizes $\sum_{i=1}^{k} |\delta(V_i)|$. Natural extensions of our structural theorem fail to hold for this variant of the hypergraph k-partitioning problem. We note that this variant is polynomial-time solvable for $k \leq 5$ (Chandrasekaran and Chekuri [12], Hirayama et al. [35], Okumoto et al. [52]), and resolving its complexity for $k \geq 6$ remains open.

We mention an open question concerning Hypergraph-k-Cut and the enumeration of min-k-cut-sets in hypergraphs for fixed k. We recall the status in graphs: the number of minimum k-partitions in a connected graph is known to be $O(n^{2k-2})$ via Karger–Stein's algorithm (Karger and Stein [40]) and $\Omega(n^k)$ via the cycle example, where n is the number of vertices; recent works improve on the upper bound to match the lower bound for fixed k. This improvement in upper bound also led to the best possible $O(n^k)$-time algorithm for Graph-k-Cut for fixed k (Gupta et al. [31–33]). For hypergraphs, the number of min-k-cut-sets is known to be $O(n^{2k-2})$ and $\Omega(n^k)$. Can we improve the upper/lower bound? Is it possible to design an algorithm for Hypergraph-k-Cut that runs in time $O(n^k p)$?

Acknowledgments

The authors thank the reviewers for carefully reading and providing constructive feedback. A preliminary version of this work appeared at the International Colloquium on Automata, Languages, and Programming 2022.

Endnotes

[1] We emphasize that the objective of Hypergraph-k-Cut is not equivalent to minimizing $\sum_{i=1}^{k} c(\delta(V_i))$.

[2] A real-valued set function $f : 2^V \to \mathbb{R}$ is submodular if $f(A) + f(B) \geq f(A \cap B) + f(A \cup B) \;\; \forall A, B \subseteq V$.

[3] An evaluation oracle for a set function $f : 2^V \to \mathbb{R}$ over a ground set V returns the value $f(S)$ given $S \subseteq V$.

[4] A real-valued set function $f : 2^V \to \mathbb{R}$ is symmetric if $f(A) = f(V \setminus A) \;\; \forall A \subseteq V$.

[5] Subhypergraph $G[A]$ has vertex set A and contains all hyperedges of G that are entirely contained within A.

[6] Suppose we know a min-k-cut-set F. Then, consider the connected components $Q_1, \ldots, Q_t$ in $G - F$ and create a partition $(P_1, \ldots, P_k)$ by taking $P_i = Q_i$ for every $i \in [k-1]$ and $P_k = \cup_{j=k}^{t} Q_j$; such a k-partition $(P_1, \ldots, P_k)$ is an optimum k-partition for Hypergraph-k-Cut.

[7] Consider the n-vertex hypergraph $G = (V, E)$ obtained from the complete bipartite graph $K_{n/2,n/2} = (L \cup R, E')$ by adding $n^2/4$ copies of two hyperedges e_1 and e_2, where $e_1 := L$ and $e_2 := R$. The minimum cut value is $n^2/4$ with (L, R) being the unique minimum cut. Moreover, for every nonempty subset $X \subsetneq L$, the 2-partition $(X, \overline{X})$ has cut value at most $n^2/2$. The cut sets of these 2-partitions are all distinct. Thus, G has exponentially many 2-approximate minimum cut sets.

References

[1] Abboud A, Krauthgamer R, Trabelsi O (2021) Subcubic algorithms for Gomory–Hu tree in unweighted graphs. *Proc. 53rd Annual ACM SIGACT Sympos. Theory Comput.* (ACM, New York), 1725–1737.
[2] Ahn K, Guha S (2009) Graph sparsification in the semi-streaming model. *Proc. 36th Internat. Colloquium Automata Languages Programming Part II* (Springer, Berlin, Heidelberg), 328–338.
[3] Ahn K, Guha S, McGregor A (2012) Analyzing graph structure via linear measurements. *Proc. 23rd Annual ACM-SIAM Sympos. Discrete Algorithms* (ACM-SIAM, New York-Philadelphia), 459–467.
[4] Ahn K, Guha S, McGregor A (2012) Graph sketches: Sparsification, spanners, and subgraphs. *Proc. 31st Sympos. Principles Database Systems* (ACM, New York), 5–14.
[5] Applegate D, Bixby R, Chvátal V, Cook W (2006) *The Traveling Salesman Problem: A Computational Study* (Princeton University Press, Princeton, NJ).
[6] Bansal N, Svensson O, Trevisan L (2019) New notions and constructions of sparsification for graphs and hypergraphs. *Proc. 60th Annual IEEE Sympos. Foundations Comput. Sci.*, 910–928.
[7] Beideman C, Chandrasekaran K, Wang W (2022) Deterministic enumeration of all minimum k-cut-sets in hypergraphs for fixed k. *Proc. 33rd Annual ACM-SIAM Sympos. Discrete Algorithms* (ACM-SIAM, New York-Philadelphia), 2208–2228.
[8] Beideman C, Chandrasekaran K, Wang W (2023) Approximate minimum cuts and their enumeration. *Sympos. Simplicity Algorithms*, 36–41.
[9] Benczur A (1995) A representation of cuts within 6/5 times the edge connectivity with applications. *Proc. 36th Annual IEEE Foundations Comput. Sci.* (IEEE, Piscataway, NJ), 92–102.
[10] Benczur A (1997) Cut structures and randomized algorithms in edge-connectivity problems. Unpublished PhD thesis, Massachusetts Institute of Technology, Cambridge, MA.
[11] Benczur A, Goemans M (2008) Deformable polygon representation and near-mincuts. Groetschel M, Katona GOH, eds. *Building Bridges: Between Mathematics and Computer Science*, Bolyai Society Mathematical Studies, vol. 19 (Springer, New York), 103–135.
[12] Chandrasekaran K, Chekuri C (2021) Min-max partitioning of hypergraphs and symmetric submodular functions. *Proc. 32nd Annual ACM-SIAM Sympos. Discrete Algorithms* (ACM-SIAM, New York-Philadelphia), 1026–1038.
[13] Chandrasekaran K, Chekuri C (2022) Hypergraph k-cut for fixed k in deterministic polynomial time. *Math. Oper. Res.* 47(4):3380–3399.
[14] Chandrasekaran K, Wang W (2023) Fixed parameter approximation scheme for min-max k-cut. *Math. Programming* 197:1093–1144.
[15] Chandrasekaran K, Zhang X (2020) Hardness of min-max hypergraph k-partition when k is part of input, Personal communication.
[16] Chandrasekaran K, Xu C, Yu X (2021) Hypergraph k-cut in randomized polynomial time. *Math. Programming* 186:85–113.
[17] Chekuri C, Li S (2020) On the hardness of approximating the k-way hypergraph cut problem. *Theory Comput.* 16(14):1–8.
[18] Chekuri C, Xu C (2018) Minimum cuts and sparsification in hypergraphs. *SIAM J. Comput.* 47(6):2118–2156.
[19] Chekuri C, Quanrud K, Xu C (2020) LP relaxation and tree packing for minimum k-cut. *SIAM J. Discrete Math.* 34(2):1334–1353.
[20] Chen Y, Khanna S, Nagda A (2020) Near-linear size hypergraph cut sparsifiers. *Proc. 61st Annual IEEE Sympo. Foundations Comput. Sci.* (IEEE, Piscataway, NJ), 61–72.
[21] Cook W, Cunningham W, Pulleyblank W, Schrijver A (1997) *Combinatorial Optimization* (Wiley, Hoboken, NJ).
[22] Dinitz EA, Karzanov AV, Lomonosov MV (1976) On the structure of a family of minimum weighted cuts in a graph. Fridman AA, ed. *Studies in Discrete Optimization* (Nauka Publishers, Moscow).
[23] Downey R, Estivill-Castro V, Fellows M, Prieto E, Rosamund F (2003) Cutting up is hard to do: The parameterised complexity of k-cut and related problems. *Electronic Notes Theoretical Comput. Sci.* 78:209–222.
[24] Fox K, Panigrahi D, Zhang F (2019) Minimum cut and minimum k-cut in hypergraphs via branching contractions. *Proc. 30th Annual ACM-SIAM Sympos. Discrete Algorithms* (ACM-SIAM, New York-Philadelphia), 881–896.
[25] Ghaffari M, Karger D, Panigrahi D (2017) Random contractions and sampling for hypergraph and hedge connectivity. *Proc. 28th Annual ACM-SIAM Sympos. Discrete Algorithms* (ACM-SIAM, New York-Philadelphia), 1101–1114.
[26] Goemans MX, Ramakrishnan VS (1995) Minimizing submodular functions over families of sets. *Combinatorica* 15(4):499–513.
[27] Goldschmidt O, Hochbaum D (1994) A polynomial algorithm for the k-cut problem for fixed k. *Math. Oper. Res.* 19(1):24–37.
[28] Guinez F, Queyranne M (2012) The size-constrained submodular k-partition problem. Accessed 2021, https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxmbGF2aW9ndWluZXpob21lcGFnZXxneDo0NDVlMThkMDg4ZWRlOGI1.

[29] Gupta A, Lee E, Li J (2018) An FPT algorithm beating 2-approximation for *k*-cut. *Proc. 29th Annual ACM-SIAM Sympos. Discrete Algorithms* (ACM-SIAM, New York-Philadelphia), 2821–2837.
[30] Gupta A, Lee E, Li J (2018) Faster exact and approximate algorithms for *k*-cut. *Proc. 59th IEEE Annual Sympo. Foundations Comput. Sci.* (IEEE, Piscataway, NJ), 113–123.
[31] Gupta A, Lee E, Li J (2019) The number of minimum *k*-cuts: Improving the Karger-Stein bound. *Proc. 51st ACM Sympos. Theory Comput.* (ACM, New York), 229–240.
[32] Gupta A, Lee E, Li J (2020) The Karger-Stein algorithm is optimal for *k*-cut. *Proc. 52nd Annual ACM Sympos. Theory Comput.* (ACM, New York), 473–484.
[33] Gupta A, Harris D, Lee E, Li J (2020) Optimal bounds for the *k*-cut problem Preprint, submitted May 17, https://arxiv.org/abs/2005.08301.
[34] Henzinger M, Williamson D (1996) On the number of small cuts in a graph. *Inform. Processing Lett.* 59(1):41–44.
[35] Hirayama T, Liu Y, Makino K, Shi K, Xu C (2023) A polynomial time algorithm for finding a minimum 4-partition of a submodular function. *Proc. 34th Annual ACM-SIAM Sympos. Discrete Algorithms* (ACM-SIAM, New York-Philadelphia), 1680–1691.
[36] Kamidoi Y, Yoshida N, Nagamochi H (2007) A deterministic algorithm for finding all minimum *k*-way cuts. *SIAM J. Comput.* 36(5):1329–1341.
[37] Kapralov M, Krauthgamer R, Tardos J, Yoshida Y (2021) Toward tight bounds for spectral sparsification of hypergraphs. *Proc. 53rd Annual ACM SIGACT Sympos. Theory Comput.* (ACM, New York), 598–611.
[38] Karger D (1993) Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. *Proc. Fourth Annual ACM-SIAM Sympos. Discrete Algorithms* (ACM-SIAM, New York-Philadelphia), 21–30.
[39] Karger D (2000) Minimum cuts in near-linear time. *J. ACM* 47(1):46–76.
[40] Karger D, Stein C (1996) A new approach to the minimum cut problem. *J. ACM* 43(4):601–640.
[41] Karlin A, Klein N, Gharan SO (2021) A (slightly) improved approximation algorithm for metric TSP. *Proc. 53rd Annual ACM SIGACT Sympos. Theory Comput.* (ACM, New York), 32–45.
[42] Kawarabayashi K, Lin B (2020) A nearly 5/3-approximation FPT algorithm for min-*k*-cut. *Proc. 31st ACM-SIAM Sympos. Discrete Algorithms* (ACM-SIAM, New York-Philadelphia), 990–999.
[43] Kawarabayashi K, Thorup M (2011) The minimum k-way cut of bounded size is fixed-parameter tractable. *Proc. 52nd Annual Sympos. Foundations Comput. Sci.* (ACM, New York), 160–169.
[44] Kogan D, Krauthgamer R (2015) Sketching cuts in graphs and hypergraphs. *Proc. 2015 Conf. Innovations Theoretical Comput. Sci.* (ACM, New York), 367–376.
[45] Lawler E (1973) Cutsets and partitions of hypergraphs. *Networks* 3(3):275–285.
[46] Li J, Panigrahi D (2020) Deterministic min-cut in poly-logarithmic max-flows. *Proc. 61st Annual Sympos. Foundations Comput. Sci.*, 85–92.
[47] Lokshtanov D, Saurabh S, Surianarayanan V (2020) A parameterized approximation scheme for Min *k*-Cut. *Proc. 61st IEEE Annual Sympos. Foundations Comput. Sci.* (IEEE, Piscataway, NJ), 798–809.
[48] Manurangsi P (2017) Almost-polynomial ratio ETH-hardness of approximating densest k-subgraph. *Proc. 49th Annual ACM Sympos. Theory Comput.* (ACM, New York), 954–961.
[49] Manurangsi P (2018) Inapproximability of maximum biclique problems, minimum *k*-cut and densest at-least-*k*-subgraph from the small set expansion hypothesis. *Algorithms* 11(1):10.
[50] Nagamochi H, Nishimura K, Ibaraki T (1997) Computing all small cuts in an undirected network. *SIAM J. Discrete Math.* 10(3):469–481.
[51] Nägele M, Sudakov B, Zenklusen R (2019) Submodular minimization under congruency constraints. *Combinatorica* 39:1351–1386.
[52] Okumoto K, Fukunaga T, Nagamochi H (2012) Divide-and-conquer algorithms for partitioning hypergraphs and submodular systems. *Algorithmica* 62(3):787–806.
[53] Quanrud K (2019) Fast and deterministic approximations for *k*-cut. *Proc. Approximation Randomization Combin. Optim. Algorithms Techniques*, vol. 23 (Springer, New York), 1–20.
[54] Queyranne M (1999) On optimum size-constrained set partitions. Presentation, Aussois Workshop on Combinatorial Optimization, Aussois, France.
[55] Raghavendra P, Steurer D (2010) Graph expansion and the unique games conjecture. *Proc. 42nd Annual ACM Sympos. Theory Comput.* (ACM, New York), 755–764.
[56] Ravi R, Sinha A (2008) Approximating *k*-cuts using network strength as a Lagrangean relaxation. *Eur. J. Oper. Res.* 186(1):77–90.
[57] Saran H, Vazirani V (1995) Finding *k* cuts within twice the optimal. *SIAM J. Comput.* 24(1):101–108.
[58] Soma T, Yoshida Y (2019) Spectral sparsification of hypergraphs. *Proc. 30th Annual ACM-SIAM Sympos. Discrete Algorithms* (ACM-SIAM, New York-Philadelphia), 2570–2581.
[59] Thorup M (2008) Minimum *k*-way cuts via deterministic greedy tree packing. *Proc. 40th Annual ACM Sympos. Theory Comput.* (ACM, New York), 159–166.
[60] Xiao M (2008) An improved divide-and-conquer algorithm for finding all minimum *k*-way cuts. *Proc. 19th Internat. Sympos. Algorithms Comput.* (ACM, New York), 208–219.
[61] Zenklusen R (2019) A 1.5-approximation for path TSP. *Proc. 30th Annual ACM-SIAM Sympos. Discrete Algorithms* (ACM-SIAM, New York-Philadelphia), 1539–1549.
[62] Zhao L (2002) Approximation algorithms for partition and design problems in networks. Unpublished PhD thesis, Kyoto University, Kyoto, Japan.
[63] Zhao L, Nagamochi H, Ibaraki T (2005) Greedy splitting algorithms for approximating multiway partition problems. *Math. Programming* 102(1):167–183.