Mathematical Programming
https://doi.org/10.1007/s10107-019-01443-7

FULL LENGTH PAPER

Series A

Hypergraph *k*-cut in randomized polynomial time

Karthekeyan Chandrasekaran[1] · **Chao Xu**[1,2] · **Xilin Yu**[1]

Received: 18 February 2018 / Accepted: 28 October 2019

Abstract

For a fixed integer $k \geq 2$, the hypergraph k-cut problem asks for a smallest subset of hyperedges whose removal leads to at least k connected components in the remaining hypergraph. While graph k-cut is solvable efficiently (Goldschmidt and Hochbaum in Math. Oper. Res. 19(1):24–37, 1994), the complexity of hypergraph k-cut has been open. In this work, we present a randomized polynomial time algorithm to solve the hypergraph k-cut problem. Our algorithmic technique extends to solve the more general hedge k-cut problem when the subgraph induced by every hedge has a constant number of connected components. Our algorithm is based on random contractions akin to Karger's min cut algorithm. Our main technical contribution is a non-uniform distribution over the hedges (hyperedges) so that random contraction of hedges (hyperedges) chosen from the distribution succeeds in returning an optimum solution with large probability. In addition, we present an alternative contraction based randomized polynomial time approximation scheme for hedge k-cut in arbitrary hedgegraphs (i.e., hedgegraphs whose hedges could have a large number of connected components). Our algorithm and analysis also lead to bounds on the number of optimal solutions to the respective problems.

Keywords Hypergraph-k-cut · Hedgegraph-k-cut · Randomized algorithm

Mathematics Subject Classification 90C27 Combinatorial Optimization

An extended abstract of this work appeared in the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018). The current version contains full proofs and cut counting results.

Chao Xu and Xilin Yu were supported in part by NSF Grants CCF-1526799 and CCF-1319376 respectively. Karthekeyan is supported by NSF Grants CCF-1907937 and CCF-1814613.

✉ Xilin Yu
yuxilin51@gmail.com

Karthekeyan Chandrasekaran
karthe@illinois.edu

Chao Xu
the.chao.xu@gmail.com

[1] University of Illinois Urbana-Champaign, Champaign, USA

[2] Present Address: Yahoo! Research, New York, USA

Published online: 13 November 2019

1 Introduction

A hypergraph is specified by a vertex set and a collection of *hyperedges*, where each hyperedge is a nonempty subset of vertices. For a fixed integer $k \geq 2$, the hypergraph k-cut problem is the following:

Hypergraph-k-Cut: Given a hypergraph with non-negative hyperedge-costs, find a least cost subset of hyperedges whose removal leads to at least k connected components in the remaining hypergraph.

Equivalently, the problem asks for a partitioning of the vertex set into k parts with minimum cost set of hyperedges crossing the partition (a hyperedge is said to cross a partition if it intersects at least two parts). This is an extension of the classic hypergraph minimum cut problem. Throughout this work, we consider k to be a fixed integer that is at least 2. Hypergraph partitioning problems were discussed as early as in 1973 by Lawler [20] and have several applications including clustering in VLSI design and network reliability (e.g., see [1,10,29,32]).

A special case of HYPERGRAPH- k- CUT in which the input is in fact a graph (i.e., all hyperedges have cardinality two) is GRAPH- k- CUT. The latter problem has been investigated thoroughly in the literature. Goldschmidt and Hochbaum gave the first polynomial time algorithm to solve GRAPH- k- CUT. Their algorithm solves $n^{\Theta(k^2)}$ minimum st-cut problems, where n is the number of vertices in the input graph [10]. Karger and Stein [17] designed a randomized algorithm for GRAPH- k- CUT that runs in time $O(n^{2(k-1)} \log^3 n)$. The deterministic running time for solving GRAPH- k- CUT has been improved over a series of works [15,16,27,28] with the fastest running in time $\tilde{O}(n^{2k-1})$ [6]. If the edges have small integer weights, then the running time has been improved further to $\tilde{O}(n^{\omega k/3})$, where ω is the matrix multiplication constant [13].

The complexity of HYPERGRAPH- k- CUT has remained an intriguing open problem since the works of [10]. The special case of $k = 2$, denoted HYPERGRAPH- 2- CUT, is well-known to admit deterministic polynomial time algorithms [18,20,21]. Several recent works have aimed at designing polynomial time algorithms but have fallen short because they are efficient or return an optimum solution only for either restricted values of k or restricted families of hypergraphs. We recall these results now. For $k = 3$, Xiao [29] showed a non-crossing structural property of a minimum solution and used it to design a polynomial time algorithm. Considering restricted input family of hypergraphs, Fukunaga [8] gave a polynomial time algorithm for HYPERGRAPH- k- CUT in constant rank hypergraphs (the *rank* of a hypergraph is the cardinality of the largest hyperedge). A randomized polynomial time algorithm for HYPERGRAPH- k- CUT in constant rank hypergraphs can also be obtained using the *uniform random contraction* technique of Karger and Stein [17] as illustrated by Kogan and Krauthgamer [19]. Thus, it has been open to determine the complexity of HYPERGRAPH- k- CUT in arbitrary rank hypergraphs for $k \geq 4$.

In this work, we present a randomized polynomial time algorithm to solve HYPERGRAPH- k- CUT in arbitrary rank hypergraphs for every fixed constant k. To the best of our knowledge, this is the first polynomial time algorithm for HYPERGRAPH- k- CUT.

Hedgegraphs Our algorithm addresses the k-cut problem in a more general graph model that has garnered much attention recently. We describe this more general model now. It is often the case with modern networks that a collection of edges in a graph are interdependent and consequently could fail together—e.g., interconnected nodes in an optical network that share and rely on a single resource. Ghaffari, Karger, and Panigrahi identified the notion of hedgegraphs as a convenient graph model for such scenarios [9]. For a vertex set V, a *hedge* is a subset of edges (not hyperedges) over the vertices in V. A *hedgegraph* $G = (V, E)$ is specified by a vertex set V and a set E of *disjoint* hedges. We do allow parallel edges, and each copy of an edge should appear in exactly one of the hedges. We will denote the *graph underlying a hedgegraph* $G = (V, E)$ to be the *multigraph* whose vertex set is V and whose edges are the union of the edges in the hedges of G. Essentially, the graph underlying a hedgegraph is a multigraph whose edges have been partitioned into hedges. In the context of modern networks, a hedgegraph is a multigraph where each hedge corresponds to a resource and a link between two nodes fails if all hedges containing an edge between the two nodes fail, i.e., all resources that the link relies upon become unavailable.

In the $s - t$ hedge cut problem (abbreviated st- HEDGE- CUT), the input is a hedgegraph with non-negative hedge costs and two specified vertices s and t. The goal is to find a least cost subset of hedges whose removal disconnects s and t in the underlying graph. In the global variant of st- HEDGE- CUT (abbreviated HEDGE- 2- CUT), the input is a hedgegraph with non-negative hedge costs and the goal is to find a least cost subset of hedges whose removal leads to at least two connected components in the underlying graph. It is known that st- HEDGE- CUT is NP-hard [30], even if each hedge consists of exactly two edges [31]. In contrast, Ghaffari et al. showed that HEDGE- 2- CUT admits a randomized polynomial time approximation scheme [9]. They also gave a quasi-polynomial time algorithm to solve HEDGE- 2- CUT. It remains open to design a polynomial time algorithm for HEDGE- 2- CUT. We make progress towards this question by addressing an interesting and non-trivial family of instances that we describe next. It will be clear that this family already encompasses hypergraphs.

The *span of a hedge* is the number of connected components in the subgraph induced by the edges in the hedge. The *span of a hedgegraph* is the largest span among its hedges. HEDGE- 2- CUT in hedgegraphs with span one reduces to HYPERGRAPH- 2- CUT (by replacing each hedge by a hyperedge over the set of vertices incident to the edges in the hedge) and is hence solvable efficiently. The complexity of HEDGE- 2- CUT for constant span hedgegraphs was raised as an open problem by Coudert et al. [7]. We generalize our techniques for HYPERGRAPH- k- CUT to design a polynomial-time algorithm for HEDGE- 2- CUT in constant span hedgegraphs. More generally, for a fixed integer $k \geq 2$, we consider the hedge k-cut problem:

Hedge-k-Cut: The input is a hedgegraph with non-negative hedge costs and the goal is to find a least cost subset of hedges whose removal leads to at least k connected components in the underlying graph.

Equivalently, the problem asks for a partitioning of the vertex set into k parts with minimum cost set of hedges crossing the partition (a hedge is said to cross a partition if it has an edge whose two end-vertices are in different parts).

1.1 Results

In the rest of the paper, we will assume that $k \geq 2$ is a fixed constant integer and avoid stating this explicitly. For a hedge e, we use $r(e)$ to denote the number of vertices incident to the edges in e. Throughout, we will denote the number of vertices and hedges in the input hedgegraph $G = (V, E)$ using $n := |V|$ and $m := |E|$ respectively, and define $M := \sum_{e \in E} r(e)$. We emphasize that M represents the input-size of hedgegraphs.

Theorem 1 *For every constant $s \geq 1$, there exists a randomized algorithm to solve* HEDGE- k- CUT *in hedgegraphs with span at most s that runs in time $O(mMn^{ks+k-s} \log n)$ and succeeds with probability at least $1 - 1/n$.*

For an input hypergraph, let n denote the number of vertices and let M denote the sum of the cardinalities of the hyperedges. By the reduction mentioned earlier, HYPERGRAPH- k- CUT reduces to HEDGE- k- CUT in span-one hedgegraphs. This reduction in conjunction with Theorem 1 immediately leads to a randomized polynomial time algorithm for HYPERGRAPH- k- CUT with running time $O(mMn^{2k-1} \log n)$. We save a factor of m on this running time by specializing the running time analysis of the same algorithm that is used in Theorem 1 for hypergraphs.

Theorem 2 *There exists a randomized algorithm to solve* HYPERGRAPH- k- CUT *that runs in time $O(Mn^{2k-1} \log n)$ and succeeds with probability at least $1 - 1/n$.*

We mention that for the special case of $k = 2$, namely HYPERGRAPH- 2- CUT, Ghaffari et al. gave an algorithm based on random contractions that runs in time $O(Mn^2 \log n)$ [9]. So, our algorithm for HYPERGRAPH- 2- CUT is slower than theirs by a factor of n. Their algorithm picks a hyperedge to contract according to a distribution that requires knowledge of the value of the optimum 2-cut. They suggest addressing this issue by a standard technique: employ a binary search to find the optimum 2-cut value. In contrast, our contraction algorithm for HYPERGRAPH- 2- CUT in Theorem 2 *does not* require knowledge of the optimum cut value and is easy to implement. Our main technical contribution is a simple and explicit distribution over the hyperedges which enables an elegant analysis of the random contraction algorithm for hypergraphs. More importantly, our algorithm generalizes naturally to resolve the complexity of the more general problem of HYPERGRAPH- k- CUT.

Next, we bound the number of minimum solutions for HYPERGRAPH- k- CUT. A set C of hyperedges in a hypergraph G is said to be a *k-cut-set* if the removal of C from G results in a hypergraph with at least k connected components. A k-cut-set in G is a *minimum k-cut-set* if its cost is equal to the minimum cost of a set of hyperedges whose removal from G results in a hypergraph with at least k connected components. Each minimum k-cut-set corresponds to a set of hyperedges crossing some partition $V_1, \ldots, V_k$ of V. Our algorithmic technique also leads to the following bound on the number of minimum k-cut-sets:

Corollary 1 *The number of distinct minimum k-cut-sets in an n-vertex hypergraph is $O(n^{2(k-1)})$.*

The bound stated in Corollary 1 generalizes and recovers (i) the bound on the number of minimum k-cut-sets in graphs by Karger and Stein [17] as well as (ii) the bound on the number of minimum 2-cut-sets in hypergraphs [5,9]. We remark that Corollary 1 counts the number of minimum k-cut-sets as opposed to the number of k-partitions that induce minimum k-cut-sets. It is well-known that the number of 2-partitions that induce minimum 2-cut-sets could be exponential in the size of the largest hyperedge [19]. For example, consider the hypergraph consisting of a single hyperedge containing all vertices where all 2-partitions induce the same minimum 2-cut-set.

Next, we address HEDGE- k- CUT in arbitrary span hedgegraphs by designing a randomized polynomial time approximation scheme. While Theorem 1 gives an algorithm to solve HEDGE- k- CUT in constant span hedgegraphs exactly in randomized polynomial-time, our next theorem gives an algorithm to obtain a $(1+\epsilon)$-approximation for HEDGE- k- CUT in arbitrary span hedgegraphs in time $n^{O(k \log(1/\epsilon))}$. For this, we generalize the techniques underlying the randomized polynomial time approximation scheme for HEDGE- 2- CUT by Ghaffari, Karger and Panigrahi [9]. A set C of hedges in a hedgegraph G is said to be a *hedge k-cut-set* if the removal of C leads to at least k connected components in the underlying graph. For $\alpha \geq 1$, a hedge k-cut-set C is said to be an *α-approximate minimum hedge k-cut-set* if the cost of C is at most α times the minimum cost of a set of hedges whose removal leads to at least k connected components in the underlying graph.

Theorem 3 *For any given $\epsilon > 0$, there exists a randomized algorithm to find a $(1+\epsilon)$-approximate minimum hedge k-cut-set that runs in time $Mn^{O(k \log(1/\epsilon))} \log n$ and succeeds with probability at least $1 - 1/n$.*

Setting ϵ to be a value that is strictly smaller than $1/\lambda$, where λ is the value of a minimum hedge k-cut-set in the input hedgegraph, we observe that a $(1+\epsilon)$-approximate minimum hedge k-cut-set would in fact be a minimum hedge k-cut-set. Thus, Theorem 3 gives a quasi-polynomial time algorithm to solve HEDGE- k- CUT (the value of λ can be found by a binary search).

Corollary 2 *There exists a randomized algorithm to solve* HEDGE- k- CUT *that runs in time $Mn^{O(k \log \lambda)} \log n$ and succeeds with probability at least $1 - 1/n$, where λ is the value of a minimum hedge k-cut-set in the input hedgegraph.*

Our algorithmic technique can also be used to bound the number of minimum k-cut-sets in hedgegraphs.

Theorem 4 *The number of distinct minimum hedge k-cut-sets in an n-vertex hedgegraph with minimum hedge k-cut-set value λ is $n^{O(k \log \lambda)}$.*

Organization. We present the preliminaries in Sect. 2, prove Theorems 1 and 2 and Corollary 1 in Sect. 3, and Theorems 3 and 4 and Corollary 2 in Sect. 4.

1.2 Related work

We begin by discussing HYPERGRAPH- k- CUT and GRAPH- k- CUT when k is part of the input (as opposed to being a fixed constant). For GRAPH- k- CUT, Goldschmidt

and Hochbaum showed that GRAPH- k- CUT is NP-hard [10] while Saran and Vazirani designed a 2-approximation algorithm [26]. Manurangsi [23] showed that there is no polynomial-time $(2-\epsilon)$-approximation for any constant $\epsilon > 0$ assuming the *Small Set Expansion Hypothesis* [25]. Gupta, Lee and Li [12] designed an algorithm that runs in time $O(2^{k^6} n^4)$ to achieve an approximation factor of $2-\delta$ for some constant $\delta > 0$. When k is part of the input, HYPERGRAPH- k- CUT is NP-hard as observed from GRAPH- k- CUT. Chekuri and Li [4] recently showed that HYPERGRAPH- k- CUT is at least as hard as the *densest k-subgraph* problem from the perspective of approximability. The densest k-subgraph problem is believed to not admit an efficient constant factor approximation assuming $P \neq NP$; it is known to not admit an efficient $n^{1/(\log\log n)^c}$-approximation for some constant $c > 0$ assuming the exponential time hypothesis [22] while the best approximation known is $O(n^{1/4+\epsilon})$ with running time $O(n^{1/\epsilon})$ for all constant $\epsilon > 0$ [2]. Chekuri and Li's result already illustrates that HYPERGRAPH- k- CUT is significantly harder to approximate than GRAPH- k- CUT when k is part of the input.

For fixed constant k, approximation algorithms for the hypergraph k-cut problem, the hypergraph k-partitioning problem, and more generally, submodular partitioning problems have been well-studied in the literature. Okumoto, Fukunaga and Nagamochi [24] reduced HYPERGRAPH- k- CUT for constant k to the node-weighted k-way cut problem in graphs[1] and thus obtained a $2(1-1/k)$-approximation. They further improved on this approximation factor for $k = 4, 5, 6$. The hypergraph k-partitioning problem is similar in flavor to the hypergraph k-cut problem but it minimizes a different objective. In the hypergraph k-partitioning problem, the input is a hypergraph and the goal is to find a partitioning of the vertex set into k non-empty parts $V_1, \ldots, V_k$ so that $\sum_{i=1}^{k} |\delta(V_i)|$ is minimum (where $\delta(V_i)$ is the set of hyperedges that cross the part V_i) [1]. The hypergraph k-partitioning problem and the hypergraph k-cut problem coincide to GRAPH- k- CUT when the input hypergraph is a graph. The hypergraph k-partitioning problem is a special case of the submodular k-partitioning problem since the hypergraph cut function is submodular. We recall that a set function $f : 2^V \to \mathbb{R}$ is submodular if $f(A \cap B) + f(A \cup B) \leq f(A) + f(B)$ for every pair of sets $A, B \subseteq V$ and is symmetric if $f(X) = f(V \setminus X)$ for all $X \subseteq V$. In the submodular k-partitioning problem, the input is a non-negative submodular set function $f : 2^V \to \mathbb{R}_+$ (given via a value oracle) and the goal is to partition the ground set V into k non-empty sets $V_1, \ldots, V_k$ so that $\sum_{i=1}^{k} f(V_i)$ is minimum. Submodular k-partitioning for the case of $k = 3$ admits an efficient algorithm [24] while approximation algorithms have been designed for larger constants k [24,33]. Submodular k-partitioning for $k = 4$ admits an efficient algorithm if the input function is symmetric [11].

We also mention that approximation algorithms are known for the variant of submodular partitioning that separates a given set of k elements [3,33]. In submodular k-way partitioning, the input is a non-negative submodular set function $f : 2^V \to \mathbb{R}_+$ (given via a value oracle) and distinct elements $v_1, \ldots, v_k \in V$, and the goal is to find a partitioning of the ground set V into k non-empty sets $V_1, \ldots, V_k$ with $v_i \in V_i$ for

[1] The node-weighted k-way cut problem is the following: Given a graph with weights on the nodes and a collection of terminal nodes, remove a smallest weight subset of non-terminal nodes so that the resulting graph has no path between the terminals.

all $i \in \{1, \ldots, k\}$ so that $\sum_{i=1}^{k} f(V_i)$ is minimum. This generalizes the hypergraph k-way cut problem (where the goal is to delete the smallest number of hyperedges in order to disconnect a given collection of k vertices in the input hypergraph). The known approximation factor for submodular k-way partitioning is 2 for general submodular functions and $3/2 - 1/k$ for symmetric submodular functions [3].

The main motivation behind the definition of hedgegraphs is to understand the connectivity properties of modern networks in which the reliability of links are correlated. In particular, the links could depend on a common resource. Two natural models of link failures have been considered in the literature: a link could fail if either all or at least one of the resources that the link depends upon fails [7]. The definition of hedgegraphs considers the former model where a link fails only if all resources that the link depends upon fail. The term *hedgegraph* for this model was given by Ghaffari, Karger and Panigrahi [9] who also showed that HEDGE- 2- CUT has a randomized polynomial time approximation scheme.

2 Preliminaries

For positive integers a and b with $a < b$, we will follow the convention that the inverse binomial expression $\binom{a}{b}^{-1}$ is 1 and $\binom{a}{b}$ is 0. The set of positive integers less than or equal to ℓ is denoted as $[\ell]$. Let $G = (V, E)$ be a hedgegraph. We will denote an edge between two vertices a and b by an unordered tuple $\{a, b\}$ and a hedge as a set of edges. We emphasize that the hedges in a hedgegraph are disjoint—if an edge appears in ℓ different hedges, then it contributes ℓ edges to the underlying multigraph. For a hedge $e \in E$, let $G[e]$ denote the subgraph induced by the edges in e. We emphasize that there are no isolated vertices in $G[e]$. Let $V(e)$ denote the vertices in $G[e]$. We recall that $r(e) = |V(e)|$. Let $s(e)$ denote the number of connected components in $G[e]$, i.e., the span of e. We note that $s(e) \geq 1$ for every non-empty hedge e. Let $s := \max\{s(e) : e \in E\}$, i.e., s denotes the span of the hedgegraph G.

Our algorithm is based on repeated contractions. Our notion of the contraction operation is identical to the well-known notion that appears in the literature. Informally, in order to contract a hedge, we contract every connected component of the hedge. We define this operation formally for the sake of completeness. Let $U \subset V$ be a subset of vertices in G. We define G/U to be a graph on vertex set $V' := (V - U) \cup \{u\}$, where u is a newly introduced vertex, and on hedge set E', where E' is obtained as follows: for each hedge $e \in E$, we define the hedge e' to be

$$e' := \{(\{a, b\} - U) \cup \{u\} : |\{a, b\} \cap U| = 1, \{a, b\} \in e\} \cup \{\{a, b\} : \{a, b\} \cap U = \emptyset, \{a, b\} \in e\}$$

and obtain $E' := \{e' : e' \neq \emptyset, e \in E\}$. For a hedge $e \in E$, let $C_1, \ldots, C_{s(e)}$ denote the vertex sets of connected components in $G[e]$. The hedgegraph obtained by contracting the hedge e, denoted G/e, is the hedgegraph obtained by contracting the vertex set of each component in $G[e]$ individually, i.e., $G/e := G/C_1/C_2/\ldots/C_{s(e)}$. We observe that G/e is invariant to the order of contraction of the components since the components are disjoint. We also observe that contracting a hedge does not increase the span.

We need the following technical lemma.

Lemma 1 (Majorization inequality, e.g., see Theorem 108 in *[14]*) *Let $y_1, \ldots, y_\ell$ and $x_1, \ldots, x_\ell$ be two finite non-increasing sequence of real numbers in $[a, b]$ with the same sum. Let $f : [a, b] \to \mathbb{R}$ be a convex function. If $\sum_{i=1}^{j} y_i \leq \sum_{i=1}^{j} x_i$ for all $1 \leq j \leq \ell$, then*

$$\sum_{i=1}^{\ell} f(y_i) \leq \sum_{i=1}^{\ell} f(x_i).$$

3 Hedge-k-cut in constant span hedgegraphs

In this section, we design an algorithm to solve HEDGE- k- CUT in constant span hedgegraphs. In Sect. 3.1, we give an outline of the algorithm for HYPERGRAPH- 2- CUT to present the main ideas underlying our algorithm for HYPERGRAPH- k- CUT. In Sect. 3.2, we present the complete algorithm and the analysis for HEDGE- k- CUT in constant span hedgegraphs. In Sect. 3.3, we improve on the running time analysis of the same algorithm by specializing it to hypergraphs and also conclude a combinatorial bound on the number of distinct minimum k-cut-sets.

3.1 Overview

We recall Karger's uniform random contraction algorithm for graphs (more generally, multigraphs where all edge costs are one): pick an edge *uniformly at random*, contract it and repeat until there are 2 vertices left at which point output the edges between the two vertices. In order to analyze the correctness, one can fix a min-cut C and argue that it is highly unlikely for the sampled edges to be in C. Indeed, suppose the value of the min-cut is λ, then every isolating cut (i.e., a cut induced by a single vertex) has value at least λ, and hence the number of edges is at least $n\lambda/2$. Consequently, the probability of picking an edge in C is at most $2/n$.

For simplicity, let us focus on the variant of HYPERGRAPH- 2- CUT where every hyperedge has cost one. Now consider the above algorithm for hypergraphs under the standard definition of hyperedge contraction (the graph G/e is obtained by removing the vertices of the hyperedge e, introducing a new vertex v and for every other hyperedge f in G that intersects e, replacing f by $(f \setminus e) \cup \{v\}$ and removing hyperedges with cardinality one). If we use the same algorithm as above to solve HYPERGRAPH- 2- CUT, then it is unclear how to analyze the resulting algorithm. This is due to the existence of n-vertex hypergraphs for which a hyperedge from a minimum cut-set could be chosen for contraction with probability as large as a constant and not at most $2/n$. We overcome this issue by choosing a hyperedge to contract from a non-uniform probability distribution.

We now present this non-uniform distribution along with the analysis. Let n be the number of vertices in the input hypergraph $G = (V, E)$. For each hyperedge $e \in E$, we define a "dampening factor" δ_e to be the probability that a uniformly random vertex

from V does *not* belong to e. We recall that $r(e)$ denotes the rank of a hyperedge e. Thus,

$$\delta_e = \frac{n - r(e)}{n}.$$

We note that $\delta_e = 0$ implies that e contains every vertex in the graph and hence is in every cut. Our algorithm for HYPERGRAPH- 2- CUT is the following: pick a hyperedge e with probability proportional to δ_e, contract and repeat until either (i) the number of vertices is at most 4 in which case, output all optimum solutions in the constant-sized current hypergraph or (ii) the dampening factor of all hyperedges is zero in which case, output all hyperedges in the current graph. The running time of the algorithm is polynomial as contraction and updating the dampening factors can be implemented in polynomial-time.

In order to analyze the correctness probability of this algorithm, let us define

$$q_n := \min_{\substack{G\text{: } n\text{-vertex hypergraph} \\ C^*\text{: } C^* \text{ is a min-cut-set in } G}} \mathsf{Pr}(\text{Algorithm outputs } C^* \text{ when executed on } G).$$

We show that $q_n \geq \binom{n}{2}^{-1}$ by induction on n. For the base case, we consider $n \leq 4$ and observe that the algorithm is correct with probability one on such n-vertex hypergraphs. For the induction step, let $n > 4$. Fix a n-vertex hypergraph $G = (V, E)$ and a min-cut-set C^* in G that together achieve the minimum for q_n. We may assume that there exists a hyperedge in G whose dampening factor is non-zero as otherwise, all hyperedges are in every cut and hence, the output is in fact an optimum and consequently, the correctness probability is one. Now, the probability that the algorithm returns C^* when executed on G is at least the probability of choosing a hyperedge $e \in E \backslash C^*$, contracting it and succeeding on the remaining hypergraph. The number of vertices in the hypergraph after contracting e is $n - r(e) + 1$. Let p_e be the probability of contracting e at this step. Hence,

$$q_n \geq \sum_{e \in E \backslash C^*} p_e q_{n-r(e)+1} = \frac{1}{\sum_{f \in E} \delta_f} \cdot \sum_{e \in E \backslash C^*} \delta_e q_{n-r(e)+1}.$$

Now, consider a hyperedge $e \in E \backslash C^*$. We have $n - r(e) \geq 1$ for otherwise the hyperedge e is in every cut and would be in C^*. Moreover, $r(e) \geq 2$ since our current hypergraph never contains a hyperedge with cardinality one. Thus, $n - r(e) + 1 \leq n - 1$. By the inductive hypothesis, we have $q_{n-r(e)+1} \geq \binom{n-r(e)+1}{2}^{-1}$. Hence,

$$\delta_e q_{n-r(e)+1} \geq \frac{n - r(e)}{n} \cdot \frac{1}{\binom{n-r(e)+1}{2}} = \frac{2}{n(n - r(e) + 1)} \geq \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}}$$

and consequently,

$$q_n \geq \frac{1}{\sum_{f\in E}\delta_f}\sum_{e\in E\setminus C^*}\frac{1}{\binom{n}{2}} = \frac{1}{\binom{n}{2}}\cdot\frac{|E\setminus C^*|}{\sum_{f\in E}\delta_f}.$$

It remains to argue that $|E\setminus C^*| \geq \sum_{f\in E}\delta_f$. By rewriting the inequality, it suffices to show that the cardinality $|C^*|$ of a min-cut-set is at most $|E| - \sum_{f\in E}\delta_f$. We will show this by an averaging argument. Consider the size of the set F_v of hyperedges containing a vertex v. Since F_v is a cut (as it isolates the vertex v), the cardinality $|C^*|$ of a min-cut-set is at most $|F_v|$ for every $v \in V$. Now, consider F_v when v is chosen uniformly. We have

$$\begin{aligned}
|C^*| \leq \mathbb{E}_{v\in V}(|F_v|) &= \sum_{f\in E}\mathsf{Pr}_{v\in V}(v \in f)\\
&= \sum_{f\in E}(1-\delta_f) \qquad\qquad \text{(by the definition of } \delta_f)\\
&= |E| - \sum_{f\in E}\delta_f.
\end{aligned}$$

This completes the proof that the algorithm succeeds with probability at least $\binom{n}{2}^{-1}$. Executing it $O(n^2\log n)$ times and returning the best answer among all executions gives an optimum solution with high probability. Our algorithm for HYPERGRAPH- k- CUT and for HEDGE- k- CUT in constant span hedgegraphs extends the above algorithm by a careful generalization of the dampening factor.

3.2 The contraction algorithm

For ease of description and analysis, we will focus on the minimum cardinality variant of HEDGE- k- CUT, i.e., when all hedges have cost one. We will specify how to adapt it to solve the minimum cost variant at the end of the section. We will present an algorithm that outputs a particular minimum hedge k-cut-set with inverse polynomial probability. Hence, returning a hedge k-cut-set with minimum value among the ones output by polynomially many executions of the contraction algorithm will indeed find a minimum hedge k-cut-set with constant probability. For the purpose of HYPERGRAPH- k- CUT, the reader should consider $s = 1$ in the following algorithm and analysis (with the standard notion of hyperedge contraction).

Let n be the number of vertices in the input hedgegraph $G = (V, E)$. For a hedge $e \in E$, we recall that $r(e)$ is the number of vertices incident to the edges in e and define

$$\delta_e := \frac{\binom{n-r(e)}{k-1}}{\binom{n}{k-1}}.$$

HEDGE-k-CUT(G):
Input: Hedgegraph $G = (V, E)$ with span s
$n \leftarrow |V|$
$\mathcal{C} \leftarrow \{E\}$ $\langle\langle \mathcal{C}$ *is the list of candidates* $\rangle\rangle$
repeat:
  if $n \leq 2(k-1)(s+1)$
    add all minimum hedge k-cut-sets in G by a brute-force search to $\mathcal{C}$
    break
  for $e \in E$ such that $\delta_e = 0$ and $|V(G/e)| \geq k$:
    add all minimum hedge k-cut-sets in G/e by a brute-force search to $\mathcal{C}$
  if $\sum_{e \in E} \delta_e = 0$
    break
  Sample an edge e in G with probability proportional to δ_e
  $G \leftarrow G/e$
  $n \leftarrow |V(G)|$
  update δ_e for every hedge e in the contracted graph G
Output all hedge k-cut-sets with minimum value among the ones in $\mathcal{C}$

Fig. 1 Contraction algorithm for constant span hedgegraphs

As per the convention established in the preliminaries, we emphasize that $\delta_e = 0$ if $n - r(e) < k - 1$. Our contraction algorithm will pick a hedge e with probability proportional to δ_e, contract it, update the values of δ_e based on the new number of vertices and $r(e)$ for every $e \in E$ and repeat until the number of vertices is small. When the number of vertices is at most a constant, we do a brute-force search. We emphasize that our brute-force search outputs all minimum hedge k-cut-sets in the hedgegraph with constant number of vertices. We do this for the purposes of convenience in the correctness analysis.

We note that a hedge e is present in every hedge k-cut-set of G if and only if $|V(G/e)| < k$. We recall that $|V(G/e)| = n - r(e) + s(e)$. Hence, if a hedge e is present in every hedge k-cut-set, then $n - r(e) + 1 \leq n - r(e) + s(e) < k$ and consequently, $\delta_e = 0$. Thus, our algorithm will never contract hedges that are present in every hedge k-cut-set. The algorithm is described in Fig. 1.

We now analyze the correctness probability of the contraction algorithm. The following lemma shows a lower bound on the number of hedges in G as a function of the minimum hedge k-cut-set value.

Lemma 2 *Let $G = (V, E)$ be a hedgegraph with $m := |E|$ and λ being the minimum hedge k-cut-set value. Then,*

$$m - \lambda \geq \sum_{e \in E} \delta_e.$$

Proof We will prove the lemma by exhibiting an upper bound on λ by the probabilistic method. Let W be a subset of $k - 1$ vertices chosen uniformly at random among all subset of vertices of size $k - 1$. Now consider the k-partition of the vertex set given by $\mathcal{P} := \{\{v\} | v \in W\} \cup \{V \setminus W\}$. We claim that the expected value of the hedge k-cut-set given by $\mathcal{P}$ is $m - \sum_{e \in E} \delta_e$ and hence $\lambda \leq m - \sum_{e \in E} \delta_e$.

We now prove the claim. Let e be a hedge in G. The probability that e does not cross $\mathcal{P}$ is $\binom{n-r(e)}{k-1}/\binom{n}{k-1} = \delta_e$. Thus, the probability that e contributes to the hedge k-cut-set $\mathcal{P}$ is $1 - \delta_e$. The claim follows by linearity of expectation. □

We need the following combinatorial statement.

Lemma 3 *Suppose $n > 2(k-1)(s+1)$. Then, for every hedge e with $r(e) \in \{2, \ldots, n-k+1\}$, we have*

$$\delta_e \binom{n - r(e) + s(e)}{(k-1)(s+1)}^{-1} \geq \binom{n}{(k-1)(s+1)}^{-1}.$$

Proof If $n - r(e) + s(e) < (k-1)(s+1)$, then $\binom{n-r(e)+s(e)}{(k-1)(s+1)}^{-1} = 1$ using the convention fixed at the beginning of Sect. 2. Since $r(e) \leq n - k + 1$, we have $\binom{n-r(e)}{k-1} \geq 1$. Thus,

$$\begin{aligned}
\delta_e &= \binom{n - r(e)}{k-1}\binom{n}{k-1}^{-1} \\
&\geq \binom{n}{k-1}^{-1} \\
&\geq \binom{n}{(k-1)(s+1)}^{-1}
\end{aligned}$$

since $n > 2(k-1)(s+1)$ and $s \geq 1$.

For the rest of the proof, we will assume that $n - r(e) + s(e) \geq (k-1)(s+1)$. We now note that the binomial in the LHS of the lemma is well-defined and non-zero. For notational convenience, let $t = (k-1)(s+1)$. Then we need to show that

$$\delta_e \binom{n - r(e) + s(e)}{t}^{-1} \geq \binom{n}{t}^{-1}. \tag{1}$$

We distinguish two cases based on whether $s + 1 \leq r(e)$ or $r(e) \leq s$.

Case 1: Suppose $s + 1 \leq r(e)$. We recall that $s \geq 1$. Since $s(e) \leq s$, we have

$$\delta_e \binom{n - r(e) + s(e)}{t}^{-1} \geq \delta_e \binom{n - r(e) + s}{t}^{-1}.$$

Let $x = r(e)$. Then it suffices to show that

$$\binom{n-x}{k-1}\binom{n}{k-1}^{-1}\binom{n-x+s}{t}^{-1} \geq \binom{n}{t}^{-1}. \tag{2}$$

Consider the LHS of (2) as a function of x. There exists a constant $C_{n,k,s}$ (that depends on n, k and s) using which the LHS can be written as

$$\begin{aligned}\text{LHS}(x) &= C_{n,k,s}\frac{(n-x)!(n-x+s-t)!}{(n-x-k+1)!(n-x+s)!}\\ &= C_{n,k,s}\frac{(n-x+s-t)!}{(n-x-k+1)!\prod_{i=1}^{s}(n-x+i)}\\ &= C_{n,k,s}\left(\frac{1}{\prod_{i=k-1}^{t-s-1}(n-x-i)}\right)\left(\frac{1}{\prod_{i=1}^{s}(n-x+i)}\right).\end{aligned}$$

The last equation follows since $t=(k-1)(s+1)$, and hence $k-1\le t-s$. From the above expression, we have that LHS(x) is an increasing function of x. Thus we only need to show inequality (2) when x is the minimum value in the domain of interest, i.e., $x=r(e)=s+1$. Hence, it suffices to show that

$$\binom{n-s-1}{k-1}\binom{n}{k-1}^{-1}\binom{n-1}{t}^{-1}\binom{n}{t}\ge 1. \tag{3}$$

To show the above, we write out the LHS of (3):

$$\begin{aligned}\text{LHS of (3)} &= \frac{(n-s-1)!(n-k+1)!}{(n-s-k)!(n-1)!(n-t)}\\ &= \frac{\prod_{i=s+1}^{s+k-1}(n-i)}{(n-t)\prod_{i=1}^{k-2}(n-i)}.\end{aligned}$$

In order to show that LHS of (3) ≥ 1, we need to show that the denominator is no greater than the numerator. Taking negative logarithm of both the denominator and the numerator, we only need to show that

$$\sum_{i=s+1}^{s+k-1}(-\log(n-i))\le\sum_{i=1}^{k-2}(-\log(n-i))-\log(n-t). \tag{4}$$

We recall that $t=(k-1)(s+1)$. We know that $\sum_{i=s+1}^{s+k-1}(n-i)=(2n-2s-k)(k-1)/2=(n-t)+\sum_{i=1}^{k-2}(n-i)$ and $\sum_{i=s+1}^{s+j}(n-i)<\sum_{i=1}^{j}(n-i)\ \forall j\in[k-2]$. Since negative logarithm is a convex function, inequality (4) follows by applying Lemma 1 using the choice $\ell:=k-1$, $y_i:=n-s-i$, $x_i:=n-i$ for every $i\in[k-2]$ and $y_{k-1}:=n-s-(k-1)$, $x_{k-1}:=n-t$.

Case 2: Suppose $r(e)\le s$. By the assumptions of the lemma, we have $r(e)\ge 2$ and hence $s\ge 2$. We recall that $r(e)$ is the number of vertices incident to the edges in e while $s(e)$ is the number of connected components in the subgraph induced by the edges in e. Hence, $r(e)-s(e)\ge r(e)/2$. Consequently, we have

$$\delta_e \binom{n - r(e) + s(e)}{t}^{-1} \geq \delta_e \binom{n - r(e)/2}{t}^{-1}.$$

Let $x = r(e)$. Then it suffices to show that

$$\binom{n-x}{k-1}\binom{n}{k-1}^{-1}\binom{n-x/2}{t}^{-1} \geq \binom{n}{t}^{-1}. \tag{5}$$

Proceeding similarly to the analysis in Case 1 above, we can show that the LHS of (5) is an increasing function of x. Then we only need to show inequality (5) for $x = 2$, i.e., we need to show that

$$\binom{n-2}{k-1}\binom{n}{k-1}^{-1}\binom{n-1}{t}^{-1}\binom{n}{t} \geq 1. \tag{6}$$

Inequality (6) is a special case of inequality (3), which we have already proven in Case 1. This concludes our proof for the combinatorial statement. □

We now show a lower bound on the success probability of the algorithm.

Theorem 5 *For an n-vertex input hedgegraph with span s, the contraction algorithm given in Fig. 1 outputs any fixed minimum hedge k-cut-set with probability at least*

$$\binom{n}{(k-1)(s+1)}^{-1}.$$

Moreover, for constant s, the contraction algorithm can be implemented to run in time $O(nmM)$, where m is the number of hedges in the input hedgegraph.

Proof For a hedgegraph H, let $\mathcal{O}(H)$ denote the set of minimum k-cut-sets in H. For $C \in \mathcal{O}(H)$, let $q(H, C)$ denote the probability that the algorithm executed on H outputs C. Let $\mathcal{G}_{n,s}$ be the set of n-vertex hedgegraphs with span at most s. We define

$$q_n := \inf_{H \in \mathcal{G}_{n,s}} \min_{C \in \mathcal{O}(H)} q(H, C).$$

We will prove by induction on n that $q_n \geq \binom{n}{(k-1)(s+1)}^{-1}$. Let $G = (V, E) \in \mathcal{G}_{n,s}$ with $C \in \mathcal{O}(G)$. Let us define $m := |E|$ and $\lambda := |C|$. We recall that the span of a hedgegraph does not increase during the execution of the algorithm.

We first note that the algorithm will terminate in finite time. This is because either the number of vertices is strictly decreasing in each iteration and consequently, the algorithm reaches the base case of $n \leq 2(k-1)(s+1)$ or satisfies the condition $\sum_{e \in E} \delta_e = 0$. If C is in the list of candidates, it will be part of the output because it is a minimum hedge k-cut-set. Therefore we just have to prove that C is in the list of candidates.

To base the induction, we consider $n \leq 2(k-1)(s+1)$. For such n, we have $q(G, C) = 1$ since the algorithm solves such instances exactly by a brute-force search and returns all minimum hedge k-cut-sets, hence $q_n = 1$.

We now show the induction step. We begin by addressing two easy cases: (i) Suppose $\delta_e = 0$ for some hedge $e \in E \backslash C$. Since $e \in E \backslash C$, we know that contracting e does not destroy C, so C is still a minimum hedge k-cut-set in G/e. We also know that $|V(G/e)| \geq k$. This is because contracting any hedge f with $|V(G/f)| < k$ would destroy all hedge k-cut-sets but C survives the contraction of e. Since $\delta_e = 0$ and $|V(G/e)| \geq k$, the algorithm will add all minimum hedge k-cut-sets in G/e including C to the list of candidates, so $q(G, C) = 1$. (ii) Suppose $C = E$. Then, all hedges are present in every hedge k-cut-set. Therefore, $\delta_e = 0$ for every hedge $e \in E$. So the algorithm executes only one iteration of the outer loop and will go to the last step. Since all hedges are present in every hedge k-cut-set, contracting any hedge $e \in E$ will destroy all hedge k-cut-sets. Consequently, the algorithm will not find any candidate other than E and the final step will correctly return all hedges in G since the initialized list contains E as a candidate. Hence, $q(G, C) = 1$.

Thus, we may assume that (i) $n > 2(k-1)(s+1)$, (ii) $\delta_e > 0$ for all $e \in E \backslash C$, and (iii) $E \backslash C \neq \emptyset$. In particular, (ii) and (iii) imply that $\sum_{e \in E} \delta_e > 0$. Let $p_e := \delta_e / \sum_{e \in E} \delta_e$ for every $e \in E$. We note that $(p_e)_{e \in E}$ is a probability distribution supported on the hedges because $p_e \geq 0$ for each $e \in E$ and $\sum_{e \in E} p_e = 1$. The algorithm picks a hedge e to contract according to the distribution defined by $(p_e)_{e \in E}$. We note that since $\sum_{e \in E} \delta_e > 0$, we have $\delta_e > 0$ for some $e \in E$ and thus, the algorithm will contract some hedge.

The algorithm executed on G outputs C if the hedge e that it contracts is in $E \backslash C$ and the algorithm executed on the contracted hedgegraph G/e outputs C. Consider then such $e \in E \backslash C$. The hedgegraph G/e has $n - r(e) + s(e) < n$ vertices and clearly, the span of G/e is at most s and hence $G/e \in \mathcal{G}_{n-r(e)+s(e),s}$. Furthermore, the k-cut-set C is still a minimum hedge k-cut-set in G/e and hence $C \in \mathcal{O}(G/e)$. Thus, $q(G/e, C) \geq q_{n-r(e)+s(e)}$ by definition. Thus, we have

$$
\begin{aligned}
q(G, C) &\geq \sum_{e \in E \backslash C} p_e \cdot q(G/e, C) \\
&\geq \sum_{e \in E \backslash C} p_e \cdot q_{n-r(e)+s(e)} \\
&= \frac{1}{\sum_{e \in E} \delta_e} \sum_{e \in E \backslash C} \delta_e \cdot q_{n-r(e)+s(e)} \\
&\geq \frac{1}{\sum_{e \in E} \delta_e} \sum_{e \in E \backslash C} \delta_e \cdot \binom{n - r(e) + s(e)}{(k-1)(s+1)}^{-1}. \qquad \text{(by induction hypothesis)}
\end{aligned}
$$

For every $e \in E \backslash C$, we know that $\delta_e > 0$, which implies that $n - r(e) \geq k - 1$ by definition of δ_e. Moreover, by the assumption on G, we have that $n > 2(k-1)(s+1)$. Hence, by Lemma 3, for every hedge $e \in E \backslash C$, we have

$$\delta_e \cdot \binom{n - r(e) + s(e)}{(k-1)(s+1)}^{-1} \geq \binom{n}{(k-1)(s+1)}^{-1}.$$

Substituting this in the previously derived lower bound for $q(G, C)$, we have

$$\begin{aligned} q(G, C) &\geq \frac{1}{\sum_{e \in E} \delta_e} \sum_{e \in E \setminus C} \binom{n}{(k-1)(s+1)}^{-1} \\ &= \frac{m - \lambda}{\sum_{e \in E} \delta_e} \binom{n}{(k-1)(s+1)}^{-1} \qquad \text{(since } |C| = \lambda \text{ and } |E| = m\text{)} \\ &\geq \binom{n}{(k-1)(s+1)}^{-1}. \qquad \text{(by Lemma 2)} \end{aligned}$$

In all cases, we have shown that $q(G, C) \geq \binom{n}{(k-1)(s+1)}^{-1}$ for an arbitrary $G \in \mathcal{G}_{n,s}$ and an arbitrary $C \in \mathcal{O}(G)$. Therefore, we have

$$q_n = \inf_{H \in \mathcal{G}_{n,s}} \min_{C \in \mathcal{O}(H)} q(H, C) \geq \binom{n}{(k-1)(s+1)}^{-1}.$$

This concludes our proof of the correctness probability by induction.

We now analyze the running time of the contraction algorithm. A hedge contraction operation takes $O(M)$ time: To contract a hedge e, we construct a table of all vertices. For each vertex, we store the corresponding vertex after contraction. The second step is contraction. We process every hedge and mark a vertex if it needs to be merged. If so, we also mark which vertex it merges to. Marking vertices takes $O(1)$ time per vertex encountered in a hedge as we only need to check the table. Therefore, marking vertices in all hedges takes $O(M)$ time in total. Then, we replace all the marked vertices with the new vertices and update the hedges accordingly in $O(M)$ time. Hence the contraction operation can be implemented to run in $O(M)$ time.

We analyze the running time of one iteration of the loop. The brute-force operation when $n \leq 2(k-1)(s+1)$ takes $O(Mk^{2(k-1)(s+1)}) = O(M)$ time. The for-loop applies at most $O(m)$ contractions. Each contraction takes $O(M)$ time and each brute-force search for minimum hedge-k-cut-set takes $O(Mk^{k+s}) = O(M)$ time. Hence the for-loop runs in time $O(mM)$. Verifying if $\sum_{e \in E} \delta_e = 0$ can be done in $O(m)$ time. Picking a random hedge given a probability distribution on the hedges takes $O(m)$ time. The last step contracts and updates the δ_e values. Contraction takes $O(M)$ time. In order to update the δ_e values, we can precompute $\binom{a}{k-1}$ for all $k - 1 \leq a \leq n$ in $O(n(k-1)) = O(n)$ arithmetic operations. After every contraction, we can thus update δ_e for each e in constant time using the table. Now, we can compute $\sum_{e \in E} \delta_e$ in $O(|E|) = O(m)$ time. With these values, the probability p_e for all $e \in E$ can be found in $O(m)$ time. Hence, the total running time of one iteration is $O(mM)$.

Since the number of vertices strictly decreases after each contraction, the total number of iterations is at most n. By the above discussion, the contraction algorithm can be implemented to run in $O(nmM)$ time. We mention that the bottleneck of the

algorithm is the for-loop. If the for-loop is never executed, then the running time is $O(nM)$. □

Theorem 1 follows from Theorem 5 by executing the contraction algorithm

$$\binom{n}{(k-1)(s+1)} \log n$$

times and outputting a hedge k-cut-set with the minimum value among all executions.

Remark 1 The contraction algorithm can be adapted to solve the minimum cost variant, where each hedge e has cost $c(e)$, and the goal is to find a subset of hedges of minimum total cost to remove so that the underlying graph has at least k connected components. In this case, we set $\delta_e := c(e)\binom{n-r(e)}{k-1}/\binom{n}{k-1}$ and run the same contraction algorithm as above. The correctness and running time arguments are analogous to the one in Theorem 5 and we avoid repeating in the interests of brevity.

3.3 Contraction algorithm for HYPERGRAPH-k-CUT

We next focus on the special case of HYPERGRAPH- k- CUT. We restate and prove Theorem 2.

Theorem 2 *There exists a randomized algorithm to solve* HYPERGRAPH- k- CUT *that runs in time* $O(Mn^{2k-1} \log n)$ *and succeeds with probability at least* $1 - 1/n$.

Proof We will show that a hypergraph can be transformed to a hedgegraph with span one without changing the value of a minimum k-cut-set. By Theorem 1, such a transformation immediately gives a randomized polynomial time algorithm to solve HYPERGRAPH- k- CUT that runs in time $O(mMn^{2(k-1)} \log n)$ and succeeds with probability at least $1 - 1/n$. We discuss the running time improvement after showing the transformation. This transformation was also mentioned in [9] to show that HYPERGRAPH- 2- CUT is a special case of HEDGE- 2- CUT.

Let $G = (V, E)$ be an input hypergraph. We construct a hedgegraph $H = (V, E')$, where E' is obtained as follows: for every hyperedge $e \in E$, fix an arbitrary vertex $v \in e$ and introduce a hedge $e' \in E'$ consisting of edges $\{v, u\}$ for all $u \in e - \{v\}$ with the cost of e' being the same as the cost of e. Thus, the subgraph induced by the edges in e', i.e., $G[e']$, is a star centered at v that is adjacent to all the vertices in e and hence has span one. We emphasize that the constructed hedges are disjoint, i.e., if an edge appears in ℓ constructed hedges, then the underlying graph has ℓ copies of the edge with each copy being present in one of the hedges.

We now show that the value of any minimum k-cut-set is preserved by this transformation. We recall that a minimum k-cut-set in a hypergraph/hedgegraph corresponds to a set of hyperedges/hedges crossing some partition of the vertex set into k non-empty parts. Let $\{V_1, \ldots, V_k\}$ denote a partitioning of the vertex set V into k non-empty parts. We claim that a hyperedge e crosses the partition $\{V_1, \ldots, V_k\}$ in the hypergraph G if and only if the corresponding hedge e' crosses the partition $\{V_1, \ldots, V_k\}$ in the hedgegraph H. Suppose e' crosses the partition $\{V_1, \ldots, V_k\}$ in the hedgegraph

H. Consider the center vertex v of e'. Without loss of generality, let $v \in V_1$. Then there exists a vertex $u \in e' \cap V_j$ for some $j \in [k]\backslash\{1\}$. Now $u, v \in e$, and hence e crosses $\{V_1, \ldots, V_k\}$ in the hypergraph G. On the other hand, suppose e crosses the partition $\{V_1, \ldots, V_k\}$ in the hypergraph G. Consider the center vertex v of the star corresponding to e'. Without loss of generality, let $v \in V_1$ and suppose e intersects V_1 and V_j for some $j \in [k]\backslash\{1\}$. Let $u \in V_j \cap e$. Then $v \in V_1$ while $u \in V_j$ and hence e' crosses $\{V_1, \ldots, V_k\}$ in the hedgegraph H.

We now prove the improved running time bound. We obtain the improvement by showing that the algorithm will never execute the for-loop for hedgegraphs with span one. For every hedge e with $\delta_e = 0$, we have that $n - r(e) < k - 1$ and consequently $|V(G/e)| = n - r(e) + s(e) = n - r(e) + 1 < k$.

This observation shows that the running time of the contraction algorithm is $O(nM)$ as analyzed in the proof of Theorem 5. The theorem now follows by running the contraction algorithm $\binom{n}{2(k-1)} \log n$ times and returning a hedge k-cut-set with minimum value among all executions. □

We now bound the number of minimum cut-sets. We restate and prove Corollary 1.

Corollary 1 *The number of distinct minimum k-cut-sets in an n-vertex hypergraph is $O(n^{2(k-1)})$.*

Proof We recall that each minimum k-cut-set is a set of edges that crosses some k-partition $V_1, \ldots, V_k$. Let $S_1, \ldots, S_\ell$ be the minimum k-cut-sets in a given n-vertex hypergraph. Let $\mathcal{E}_i$ be the indicator variable that S_i survives the contraction algorithm until there are at most $4(k-1)$ vertices. Then, by Theorem 5 (by considering $s = 1$ for hypergraphs), we have that the expected value of $\mathcal{E}_i$ is at least $\binom{n}{2(k-1)}^{-1}$. The number of possible minimum k-cut-sets in a hypergraph with $4(k-1)$ vertices is bounded above by the number of k-partitions, which is at most $k^{4(k-1)}$. Hence, $\sum_{i=1}^{\ell} \mathrm{E}(\mathcal{E}_i) \leq k^{4(k-1)}$. Hence, $\ell \leq k^{4(k-1)} \binom{n}{2(k-1)} = O(n^{2(k-1)})$. □

4 RPTAS for HEDGE-k-CUT

In this section, we provide a randomized polynomial time approximation scheme and a quasi-polynomial time exact algorithm for HEDGE- k- CUT for constant k. We generalize the contraction approach for HEDGE- 2- CUT given by Ghaffari, Karger and Panigrahi [9]. Their contraction algorithm distinguishes large and small hedgegraphs based on the existence of small, medium, and large hedges. We generalize these definitions for the purposes of HEDGE- k- CUT and handle the cases appropriately. We again focus on the minimum cardinality variant throughout this section. The algorithm and the analysis can be adapted for the minimum cost variant similar to Remark 1.

Let $G = (V, E)$ be a hedgegraph. We define a hedge e to be *small* if $r(e) < n/(4(k-1))$, *moderate* if $n/(4(k-1)) \leq r(e) < n/(2(k-1))$, and *large* if $r(e) \geq n/(2(k-1))$. We define a hedgegraph to be *large* if it contains at least one large hedge, and to be *small* otherwise. We use the algorithm in Fig. 2.

The following lemma bounds the number of branching steps performed by the algorithm.

CONTRACT(G):
Input: Hedgegraph $G = (V, E)$
if G has k vertices
  return $F = E$
else if the graph underlying G has at least k components
  return $F = \emptyset$
$n \leftarrow |V|$
$F \leftarrow \{e \mid n - r(e) + s(e) \leq k - 1, e \in E\}$
$G \leftarrow G - F$
if G is a small hedgegraph
  Sample a hedge e uniformly at random from E
  $H \leftarrow G/e$
  return $F \cup$ CONTRACT(H)
$L \leftarrow$ set of large and moderate hedges in G
Sample a uniform random bit b
if $b = 0$ $\langle\langle$*Branch (a)*$\rangle\rangle$
  $H_1 \leftarrow G - L$
  return $F \cup L \cup$ CONTRACT(H_1)
else $\langle\langle$*Branch (b)*$\rangle\rangle$
  Sample a hedge e uniformly at random from L
  $H_2 \leftarrow G/e$
  return $F \cup$ CONTRACT(H_2).

Fig. 2 Contraction algorithm for arbitrary span hedgegraphs

Lemma 4 *The total number of branching steps in one execution of the contraction algorithm on an n-vertex hedgegraph is at most*

$$\log_{\frac{8(k-1)}{8(k-1)-1}} n.$$

Proof We prove this lemma by induction on n. Let $G = (V, E)$ be an n-vertex hedgegraph. We consider $n = k$ as base case. For such n, the algorithm terminates without a branching step, so the statement is true.

We now show the induction step. If the graph underlying G has at least k components, then the algorithm terminates without a branching step, so the statement is again true.

If G is small, then step 4 of the algorithm is performed with a recursive call on a hedgegraph with strictly fewer vertices. Therefore, the inductive hypothesis can be applied and it directly implies the statement since no branching step is performed.

If G is large, then we go to either branch (a) or branch (b). In branch (a), the algorithm recurses on a small hedgegraph H_1. Since the number of vertices spanned by each hedge never increases during the execution of the algorithm, the hedgegraph H_1 will not become a large hedgegraph, and hence will not encounter another branching, until its number of vertices is halved. Then by the inductive hypothesis, the total number of branchings in the algorithm is at most

$$1 + \log_{\frac{8(k-1)}{8(k-1)-1}} \left(\frac{n}{2}\right) \leq \log_{\frac{8(k-1)}{8(k-1)-1}} n.$$

We have proved the induction step for branch (a). Next we will show the induction step for branch (b). In branch (b), let e be the contracted hedge. Then the algorithm recurses on the hedgegraph H_2 which has $n-r(e)+s(e)$ vertices. Since each connected component of any hedge has at least two vertices, we have $s(e) \leq r(e)/2$, and thus $n - r(e) + s(e) \leq n - r(e)/2$. Since e is either a large or a moderate hedge, we have that $r(e) \geq n/(4(k-1))$, and hence the hedgegraph H_2 has at most $n - r(e)/2 \leq n \cdot (1 - 1/(8(k-1))) = n \cdot (8(k-1) - 1)/(8(k-1))$ vertices. Therefore, by the inductive hypothesis, the total number of branchings in the algorithm is at most

$$1 + \log_{\frac{8(k-1)}{8(k-1)-1}} \left(\frac{8(k-1)-1}{8(k-1)} \cdot n \right) = \log_{\frac{8(k-1)}{8(k-1)-1}} n.$$

□

We next show that the algorithm always outputs a feasible solution and that it can be implemented to run in polynomial-time.

Lemma 5 *The contraction algorithm given in Fig. 2 always outputs a hedge k-cut-set. Moreover, the algorithm can be implemented to run in time $O(Mn)$.*

Proof We first note that any hedge e with $n - r(e) + s(e) \leq k - 1$ must be in every hedge k-cut-set. By deleting such hedges from the input hedgegraph and adding them to the output set F, the algorithm ensures that it never contracts hedges such that the resulting hedgegraph has at most $k - 1$ components (vertices). So, the algorithm always outputs a hedge k-cut-set.

Next, we show that the contraction algorithm can be implemented to run in $O(Mn)$ time. In the contraction algorithm, finding the set of hedges e with $n - r(e) + s(e) \leq k - 1$ and finding the set of large and moderate hedges can each be done in $O(m)$ time. Similar to the proof of Theorem 5, a contraction step can be implemented to run in $O(M)$ time by processing hedges one by one to mark contracted vertices and replacing them with a new vertex. Since in one execution of the contraction algorithm there can be at most n contractions and $O(\log n)$ branching steps by Lemma 4, the contraction algorithm can be implemented to run in $O(Mn)$ time. □

Next, we state a few helper lemmas which will be used to lower bound the success probability of the algorithm in returning a $(1+\epsilon)$-approximate minimum hedge k-cut-set. We recall that for a hedge e, the set of vertices incident to the edges in e is denoted by $V(e)$ and the number of vertices incident to the edges in e is denoted by $r(e)$.

Lemma 6 *Let $G = (V, E)$ be a hedgegraph with minimum hedge k-cut-set value λ. Let W be a subset of $k-1$ vertices and let $E(W) := \{e \in E : |V(e) \cap W| \geq 1\}$. Then*

$$|E(W)| \geq \lambda.$$

Proof Let $\mathcal{P} = \{\{v\} | v \in W\} \cup \{V \backslash W\}$ be the k-partitioning of the vertex set induced by W. Let $E(\mathcal{P})$ be the set of hedges that cross $\mathcal{P}$. Therefore, it is a hedge k-cut-set. Since each hedge contains at least two vertices, every hedge crossing $\mathcal{P}$ must contain at least one vertex in W. Therefore, $|E(W)| \geq |E(P)| \geq \lambda$. □

Lemma 7 *Let $G = (V, E)$ be an n-vertex hedgegraph with minimum hedge k-cut-set value λ. Then,*

$$\sum_{e \in E} r(e) \geq \frac{n\lambda}{k-1}.$$

Proof Let $\mathcal{U}$ denote the set of all subsets of vertices of size $k-1$. Then $|\mathcal{U}| = \binom{n}{k-1}$. For a subset W of $k-1$ vertices, let $E(W)$ be the same as defined in Lemma 6. For a vertex v, we use $\deg(v)$ to denote the number of hedges in E which have edges incident to v.

We observe that $\sum_{W \in \mathcal{U}} \sum_{v \in W} \deg(v) = \binom{n-1}{k-2} \sum_{v \in V} \deg(v)$. Indeed, for each $v \in V$, the term $\deg(v)$ appears in the LHS whenever $v \in W$. There are $\binom{n-1}{k-2}$ sets W in $\mathcal{U}$ that contain v, so the observation follows. Therefore,

$$\begin{aligned}
\sum_{e \in E} r(e) &= \sum_{v \in V} \deg(v) \\
&= \binom{n-1}{k-2}^{-1} \sum_{W \in \mathcal{U}} \sum_{v \in W} \deg(v) \qquad \text{(by the above observation)} \\
&= \binom{n-1}{k-2}^{-1} \sum_{W \in \mathcal{U}} \sum_{e \in E} |V(e) \cap W| \\
&= \binom{n-1}{k-2}^{-1} \sum_{W \in \mathcal{U}} |E(W)| \qquad \text{(by definition of } E(W)) \\
&\geq \binom{n-1}{k-2}^{-1} \sum_{W \in \mathcal{U}} \lambda \qquad \text{(by Lemma 6)} \\
&= \binom{n-1}{k-2}^{-1} \binom{n}{k-1} \lambda \\
&= \frac{n\lambda}{k-1}.
\end{aligned}$$

□

Lemma 8 *If $G = (V, E)$ is an n-vertex m-hedge small hedgegraph with C being a minimum hedge k-cut-set in G with value λ, then*

(i) $\sum_{e \in E \setminus C} r(e) \geq n\lambda/(2(k-1))$ *and*
(ii) $m \geq 2\lambda$.

Proof Since C contains λ hedges, and each hedge $e \in E$ has $r(e) \leq n/(2(k-1))$, we have that $\sum_{e \in C} r(e) \leq n\lambda/2(k-1)$. Hence, by Lemma 7, we have that $\sum_{e \in E \setminus C} r(e) \geq n\lambda/(2(k-1))$.

Moreover, $\sum_{e \in E} r(e) \leq m(n/2(k-1))$ since every hedge e has $r(e) \leq n/(2(k-1))$. Again, by Lemma 7, we have that $m \geq 2\lambda$. □

Lemma 9 *For every $x \in (0, 1/2)$ and $c \geq 4$, we have $(1-x) \cdot (1-x/c)^{-3c/2} \geq 1$.*

Proof Let $f(x) := (1-x) \cdot (1-x/c)^{-3c/2}$. Then

$$f'(x) = \left(1 - \frac{x}{c}\right)^{-\frac{3c}{2}} \cdot \left(\frac{3c(1-x)}{2(c-x)} - 1\right).$$

Since the first factor $(1-x/c)^{-3c/2} > 0$ for all $x \in (0, 1/2), c \geq 4$, the sign of $f'(x)$ depends on the sign of $3c(1-x)/(2(c-x)) - 1$. If we solve $f'(x) = 0$ for x, we have $x = c/(3c-2)$. Since $c \geq 4$, we have that $2/c \leq 1$ and hence $c/(3c-2) = 1/(3-2/c) \leq 1/2$. Then $c/(3c-2)$ divides the interval $(0, 1/2)$ into two pieces and we consider the sign of $f'(x)$ in these two pieces separately. When $0 < x < c/(3c-2)$,

$$\frac{3c(1-x)}{2(c-x)} - 1 = \frac{c - x(3c-2)}{2(c-x)} > 0.$$

When $x > c/(3c-2)$,

$$\frac{c - x(3c-2)}{2(c-x)} < 0.$$

Therefore, we know $f'(x) \geq 0$ for $x \in (0, c/(3c-2)]$ and $f'(x) < 0$ for $x \in (c/(3c-2), 1/2)$. Since $f(0) = 1$, $f(x) \geq 1$ for $x \in (0, c/(3c-2)]$. Now we only need to show that $f(x) \geq 1$ for $x \in (c/(3c-2), 1/2)$. Since $f'(x) < 0$ for $x \in (c/(3c-2), 1/2)$, we only need to show that $f(x) \geq 1$ when $x = 1/2$. We have $f(1/2) = 1/2 \cdot (1 - 1/(2c))^{-3c/2}$, which is a decreasing function of c for $c \geq 1/2$. We note that $\lim_{c \to \infty} 1/2 \cdot (1 - 1/(2c))^{-3c/2} = e^{3/4}/2 > 1$, and hence we have $f(1/2) > 1$. □

We now lower bound the success probability of the algorithm.

Lemma 10 *For an n-vertex input hedgegraph, the contraction algorithm given in Fig. 2 outputs a $(1+\epsilon)$-approximate minimum hedge k-cut-set with probability at least $n^{-O(k \log(1/\epsilon))}$.*

Proof Let $\mathcal{H}(n, \ell)$ be the family of hedgegraphs on n vertices for which the contraction algorithm will always terminate using at most ℓ branchings. We say that the algorithm *succeeds* on an input hedgegraph H if it outputs a $(1+\epsilon)$-approximate minimum hedge k-cut-set of H. Let $q(H)$ denote the probability that the algorithm succeeds on H. We define

$$q_{n,\ell} := \inf_{H \in \mathcal{H}(n,\ell)} q(H).$$

For notational simplicity, let $\gamma := \epsilon/(1+\epsilon)$. We will first prove that

$$q_{n,\ell} \geq n^{-6(k-1)} \cdot \left(\frac{\gamma}{2}\right)^{\ell} \quad \forall\, n \geq k. \tag{7}$$

Let $G \in \mathcal{H}(n, \ell)$, with vertex set $V = [n]$ and hedge set E. Let $m := |E|$. Let us fix a minimum hedge k-cut-set C of G and suppose that its value is λ. We distinguish three cases and handle them differently.

1. Suppose G is small. We prove statement (7) by induction on n. To base the induction, we consider $n = k$. Then, G has only one hedge k-cut-set and the algorithm returns it. So $q(G) = 1$ and hence $q_{n,\ell} = 1$.

   We next show the induction step. If the graph underlying G has at least k components, then, the empty set is the only minimum hedge k-cut-set and the algorithm returns it. Hence, $q(G) = 1$. Thus, we may assume that $n > k$ and the graph underlying G has fewer than k components.

   The algorithm succeeds on G if it contracts a hedge e that is not in C and the algorithm succeeds on the resulting hedgegraph G/e which has $n - r(e) + s(e)$ vertices. Hence, $q(G) \geq 1/m \cdot \sum_{e \in E \setminus C} q(G/e)$. We note that $G/e \in \mathcal{H}(n - r(e) + s(e), \ell)$ and that $n - r(e) + s(e) \leq n - r(e)/2$ for any hedge e. Therefore,

$$\begin{aligned}
q(G) &\geq \frac{1}{m} \sum_{e \in E \setminus C} q(G/e) \\
&\geq \frac{1}{m} \sum_{e \in E \setminus C} q_{n-r(e)+s(e),\ell} \qquad \text{(by definition of } q_{n,l}) \\
&\geq \frac{1}{m} \sum_{e \in E \setminus C} (n - r(e) + s(e))^{-6(k-1)} \cdot \left(\frac{\gamma}{2}\right)^{\ell} \qquad \text{(by inductive hypothesis)} \\
&\geq \frac{1}{m} \sum_{e \in E \setminus C} \left(n - \frac{r(e)}{2}\right)^{-6(k-1)} \cdot \left(\frac{\gamma}{2}\right)^{\ell} \\
&= \frac{m - \lambda}{m} \left(\frac{\gamma}{2}\right)^{\ell} \frac{1}{m - \lambda} \sum_{e \in E \setminus C} \left(n - \frac{r(e)}{2}\right)^{-6(k-1)}.
\end{aligned}$$

   Since $k \geq 2$, and $n - r(e)/2 \geq 1$ for all hedges e, the function $f(r(e)) := (n - r(e)/2)^{-6(k-1)}$ is convex as a function of $r(e)$ for every hedge $e \in E$. By Jensen's inequality, we obtain

$$\frac{1}{m - \lambda} \sum_{e \in E \setminus C} \left(n - \frac{r(e)}{2}\right)^{-6(k-1)} \geq \left(n - \frac{\sum_{e \in E \setminus C} r(e)}{2(m - \lambda)}\right)^{-6(k-1)}.$$

   Therefore, we have

$$q(G) \geq \frac{m - \lambda}{m} \cdot \left(\frac{\gamma}{2}\right)^{\ell} \cdot \left(n - \frac{\sum_{e \in E \setminus C} r(e)}{2(m - \lambda)}\right)^{-6(k-1)}$$

$$
\begin{aligned}
&\geq \frac{m-\lambda}{m} \cdot \left(\frac{\gamma}{2}\right)^{\ell} \cdot \left(n - \frac{n\lambda/(2(k-1))}{2m}\right)^{-6(k-1)} && \text{(by Lemma 8)} \\
&= (1-\frac{\lambda}{m}) \left(\frac{\gamma}{2}\right)^{\ell} n^{-6(k-1)} \left(1 - \frac{\lambda}{4(k-1)m}\right)^{-6(k-1)} \\
&= \left(\frac{\gamma}{2}\right)^{\ell} n^{-6(k-1)} \cdot (1-x) \left(1 - \frac{x}{4(k-1)}\right)^{-6(k-1)}.
\end{aligned}
$$

The last equality follows by setting $x := \lambda/m$. We have $x \in (0, 1/2)$ since $m \geq 2\lambda$ by Lemma 8. We recall that we would like to prove that $q(G) \geq n^{-6(k-1)} \cdot (\gamma/2)^{\ell}$, so we only need to prove that $(1-x)\,(1-x/(4(k-1)))^{-6(k-1)} \geq 1$ for $x \in (0, 1/2)$. By setting $c = 4(k-1)$ in Lemma 9 it indeed holds.

2. Suppose G is large and $|L\backslash C| \geq \gamma \cdot |L|$. We prove statement (7) by induction on n. The base case of $n = k$ and the inductive case where the graph underlying G has at least k components can be handled the same way as in case 1. Therefore, we may again assume that $n > k$ and the graph underlying G has fewer than k components.

   The algorithm succeeds if it goes to branch (b), contracts a hedge $e \in L\backslash C$, and succeeds on the resulting graph $H_2 = G/e$. By the condition that $|L\backslash C| \geq \gamma \cdot |L|$, the probability of picking a hedge $e \in L\backslash C$ to contract is at least γ. Hence, $q(G) \geq 1/2 \cdot \gamma \cdot q(H_2)$. The algorithm contracts either a moderate or a large hedge e for which $r(e) \geq n/(4(k-1))$ and H_2 has $n - r(e) + s(e)$ vertices where $n - r(e) + s(e) \leq n - r(e)/2 \leq n - 1$. We also note that $H_2 \in \mathcal{H}(|V(H_2)|, \ell - 1)$. Therefore,

$$
\begin{aligned}
q(G) &\geq \frac{1}{2} \cdot \gamma \cdot q(H_2) \\
&\geq \frac{1}{2} \cdot \gamma \cdot q_{|V(H_2)|,\ell-1} && \text{(by definition of } q_{n,l}) \\
&\geq \frac{1}{2} \cdot \gamma \cdot (|V(H_2)|)^{-6(k-1)} \cdot \left(\frac{\gamma}{2}\right)^{\ell-1} && \text{(by inductive hypothesis)} \\
&\geq \frac{1}{2} \cdot \gamma \cdot n^{-6(k-1)} \left(\frac{\gamma}{2}\right)^{\ell-1} \\
&= n^{-6(k-1)} \cdot \left(\frac{\gamma}{2}\right)^{\ell}.
\end{aligned}
$$

3. Suppose G is large and $|L\backslash C| < \gamma \cdot |L|$. We will prove that $q(G) \geq n^{-6(k-1)} \cdot (\gamma/2)^{\ell}$ by induction on ℓ. The following claim shows the base case of the statement, i.e., for $\ell = 0$:

Claim $q(G) \geq n^{-6(k-1)}$ for all $n \geq k$ and $G \in \mathcal{H}(n, 0)$.

Proof We prove by induction on n. For the base case where $n = k$, we have $q(G) = 1 \geq n^{-6(k-1)}$ for all $G \in \mathcal{H}(n, 0)$. For the inductive step, consider $G \in \mathcal{H}(n, 0)$ to be a hedgegraph on $n > k$ vertices and m hedges with a fixed minimum hedge k-cut-set C in G. We note that G is small since $G \in \mathcal{H}(n, 0)$, therefore we are in case 1. Hence,

we have

$$q(G) \geq n^{-6(k-1)} \cdot \left(\frac{\gamma}{2}\right)^{\ell} = n^{-6(k-1)}.$$

by the same proof as that of case 1. □

Next we show the inductive step where $l \geq 1$. If $n = k$ or if the graph underlying G has at least k components, the algorithm again succeeds on G with probability 1 as argued in case 1 and thus we may assume that $n > k$ and G has less than k components. We will show that the algorithm succeeds if it goes to branch (a) and succeeds on the resulting graph $H_1 = G - L$. Suppose the algorithm follows branch (a). Since $|L \backslash C| < \gamma \cdot |L|$, we have $|L \cap C| = |L| - |L \backslash C| > (1-\gamma) \cdot |L|$. Then, the value of a minimum hedge k-cut-set in H_1 is at most $|C - L| = |C| - |L \cap C| < |C| - (1-\gamma)|L|$. If the recursive call on H_1 succeeds, then the hedge k-cut-set it returns is of size at most $(1+\epsilon)(|C| - |L|(1-\gamma)) = |C|(1+\epsilon) - |L|$. Consequently, the algorithm on G returns a hedge k-cut-set of size at most $(|C|(1+\epsilon) - |L|) + |L| = |C|(1+\epsilon)$, i.e., a $(1+\epsilon)$-approximate minimum hedge k-cut-set of G. By definition, $H_1 \in \mathcal{H}(n, \ell-1)$. Clearly, the algorithm goes to branch (a) with probability $1/2$. Therefore, by the above argument and the inductive hypothesis,

$$q(G) \geq \frac{1}{2} \cdot q(H_1) \geq \frac{1}{2} \cdot q_{n,\ell-1} \geq \frac{1}{2} \cdot \left(\frac{\gamma}{2}\right)^{\ell-1} \cdot n^{-6(k-1)} \geq \left(\frac{\gamma}{2}\right)^{\ell} \cdot n^{-6(k-1)}.$$

In all cases, we have shown that $q(G) \geq (\gamma/2)^{\ell} \cdot n^{-6(k-1)}$ for an arbitrary $G \in \mathcal{H}(n, \ell)$. Therefore, for all $n \geq k$, we have

$$q_{n,\ell} = \inf_{H \in \mathcal{H}(n,\ell)} q(H) \geq n^{-6(k-1)} \cdot \left(\frac{\gamma}{2}\right)^{\ell}.$$

We know that $\gamma < 1$. Substituting the upper bound on ℓ from Lemma 4, we obtain that

$$q_{n,\ell} \geq n^{-6(k-1)} \left(\frac{\gamma}{2}\right)^{\ell} \geq n^{-6(k-1)} \left(\frac{\gamma}{2}\right)^{\log_{\frac{8(k-1)}{8(k-1)-1}} n}.$$

If $\epsilon \geq 1$, then $\gamma \geq 1/2$, so $q_{n,\ell}$ is at least inverse polynomial in n. If $\epsilon < 1$, then $\gamma = \epsilon/(1+\epsilon) > \epsilon/2$. So the success probability is at least

$$n^{-6(k-1)} \left(\frac{\epsilon}{4}\right)^{\log_{\frac{8(k-1)}{8(k-1)-1}} n} = n^{-O(k \log \frac{1}{\epsilon})}.$$

□

Theorem 3 follows from Lemmas 10 and 5 by executing the contraction algorithm $n^{O(k \log 1/\epsilon)} \log n$ times and returning a hedge k-cut-set with the minimum value among all executions.

Next, in order to prove Theorem 4, we bound the probability that the contraction algorithm returns any fixed minimum hedge k-cut-set. We note that the following

result cannot be obtained by setting $\epsilon < 1/\lambda$ in Lemma 10 since Lemma 10 only lower bounds the probability that *some* $(1+\epsilon)$-approximate minimum hedge k-cut-set is returned by the algorithm while the following result lower bounds the probability that *any* specific minimum hedge k-cut-set is returned by the algorithm. The proof structure of the following lemma is similar to that of Lemma 10. However, we point out that the condition on which we distinguish the second and third cases (and consequently the proof of these two cases) are different from that of Lemma 10.

Lemma 11 *For an n-vertex input hedgegraph, the contraction algorithm given in Fig. 2 outputs any fixed minimum hedge k-cut-set with value λ with probability $n^{-O(k \log \lambda)}$.*

Proof For a hedgegraph H, let $\mathcal{O}(H)$ denote the set of minimum k-cut-sets in H. Let $\mathcal{H}(n, \ell)$ be the family of hedgegraphs on n vertices for which the contraction algorithm will always terminate using at most ℓ branchings. Let $\mu(H, C)$ denote the probability that the algorithm returns C on H. We define

$$\mu_{n,\ell} := \inf_{H \in \mathcal{H}(n,\ell)} \min_{C \in \mathcal{O}(H)} \mu(H, C).$$

We will prove that

$$\mu_{n,\ell} \geq n^{-6(k-1)} \cdot \left(\frac{1}{2(1+\lambda)}\right)^{\ell} \quad \forall\, n \geq k.$$

Let $G \in \mathcal{H}(n, \ell)$, with vertex set $V = [n]$ and hedge set E. Let $m := |E|$. Let us fix a minimum hedge k-cut-set C of G and suppose that its value is λ. We again distinguish three cases. In each case, arguments identical to that in the proof of Lemma 10 address the case of $n = k$ and when the graph underlying G has at least k components. So, we may again assume that G has greater than k vertices and fewer than k components.

1. Suppose G is small. The arguments for this case are identical to that of the first case in the proof of Lemma 10. We avoid repeating in the interests of brevity.
2. Suppose G is large and $|L \backslash C| \geq 1$. We will prove the statement by induction on n. The algorithm returns C on G if it goes to branch (b), contracts a hedge $e \in L \backslash C$, and returns C on the resulting graph $H_2 = G/e$. The probability of picking a hedge $e \in L \backslash C$ to contract is $|L \backslash C|/|L|$. Since $|L \backslash C| \geq 1$ and moreover $|L \cap C| \leq |C| = \lambda$, we have $|L \backslash C|/|L| \geq 1/(1+\lambda)$. The algorithm contracts either a moderate or a large hedge e for which $r(e) \geq n/(4(k-1))$ and H_2 has $n - r(e) + s(e)$ vertices where $n - r(e) + s(e) \leq n - r(e)/2$. Hence, H_2 has at most $n - r(e)/2 \leq n - 1$ vertices. We also note that $H_2 \in \mathcal{H}(|V(H_2)|, \ell - 1)$ and if the hedge e that is contracted to obtain H_2 is not in C, then $C \in \mathcal{O}(H_2)$.

Therefore,

$$\begin{aligned}\mu(G, C) &\geq \frac{1}{2} \cdot \frac{1}{(1+\lambda)} \cdot \mu(H_2, C) \\ &\geq \frac{1}{2(1+\lambda)} \cdot \mu_{|V(H_2)|,\ell-1} \quad \text{(by definition of } \mu_{n,l}) \\ &\geq (|V(H_2)|)^{-6(k-1)} \cdot \left(\frac{1}{2(1+\lambda)}\right)^{\ell} \quad \text{(by inductive hypothesis)} \\ &\geq n^{-6(k-1)} \cdot \left(\frac{1}{2(1+\lambda)}\right)^{\ell}.\end{aligned}$$

3. Suppose G is large and $L \subseteq C$. In this case, we will prove the statement by induction on ℓ. The algorithm returns C on G if it goes to branch (a) and returns $C - L$ on $H_1 = G - L$. We note that $H_1 \in \mathcal{H}(n, \ell - 1)$ and $C \backslash L \in \mathcal{O}(H_1)$. Therefore,

$$\mu(G, C) \geq \frac{1}{2} \cdot \mu(H_1, C \backslash L) \geq \frac{1}{2} \cdot \mu_{n,\ell-1}.$$

   By induction on ℓ, we may now show that $\mu(G, C) \geq n^{-6(k-1)} \cdot (1/2(1+\lambda))^{\ell}$. The rest of the proof is identical to that of case 3 of the proof of Lemma 10, so we avoid repeating it.

In all cases, we have shown that $\mu(G, C) \geq n^{-6(k-1)}(1/2(1+\lambda))^{\ell}$ for an arbitrary $G \in \mathcal{H}(n, \ell)$ and $C \in \mathcal{O}(G)$. Therefore, for all $n \geq k$, we have

$$\mu_{n,\ell} \geq n^{-6(k-1)} \cdot \left(\frac{1}{2(1+\lambda)}\right)^{\ell}.$$

Substituting the upper bound on ℓ from Lemma 4 gives

$$\mu_{n,\ell} \geq n^{-6(k-1)} \cdot \left(\frac{1}{2(1+\lambda)}\right)^{\log_{\frac{8(k-1)}{8(k-1)-1}} n} = n^{-O(k \log \lambda)}.$$

□

Lemmas 11 and 5 lead to Corollary 2 by executing the contraction algorithm $n^{O(k \log \lambda)} \log n$ times and returning a hedge k-cut-set with the minimum value among all executions (the value λ can be found by a binary search). Lemma 11 also leads to Theorem 4 similar to the proof of Corollary 1 from Lemma 5.

References

1. Alpert, C., Kahng, A.: Recent developments in netlist partitioning: a survey. Integr. VLSI J. **19**(1–2), 1–81 (1995)

2. Bhaskara, A., Charikar, M., Chlamtac, E., Feige, U., Vijayaraghavan, A.: Detecting high log-densities: an $o(n^{\frac{1}{4}})$ approximation for densest k-subgraph. In: Proceedings of the 42nd Annual ACM Symposium on Theory of Computing, STOC '10, pp. 201–210 (2010)
3. Chekuri, C., Ene, A.: Approximation algorithms for submodular multiway partition. In: Proceedings of the 52nd IEEE Annual Symposium on Foundations of Computer Science, FOCS '11, pp. 807–816 (2011)
4. Chekuri, C., Li, S.: A note on the hardness of approximating the k-way hypergraph cut problem, Manuscript, http://chekuri.cs.illinois.edu/papers/hypergraph-kcut.pdf (2015)
5. Chekuri, C., Xu, C.: Computing minimum cuts in hypergraphs. In: Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '17, pp. 1085–1100 (2017)
6. Chekuri, Chandra, Quanrud, Kent, Xu, Chao.: LP relaxation and tree packing for minimum k-cuts. In: 2nd Symposium on Simplicity in Algorithms (SOSA 2019), pp. 7:1–7:18 (2019)
7. Coudert, D., Datta, P., Perennes, S., Rivano, H., Voge, M.-E.: Shared Risk Resource Group: Complexity and Approximability Issues, Research Report RR-5859, INRIA (2006)
8. Fukunaga, T.: Computing minimum multiway cuts in hypergraphs. Discrete Optim. **10**(4), 371–382 (2013)
9. Ghaffari, M., Karger, D., Panigrahi, D.: Random Contractions and Sampling for Hypergraph and Hedge Connectivity. In: Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '17, pp. 1101–1114 (2017)
10. Goldschmidt, O., Hochbaum, D.: A polynomial algorithm for the k-cut problem for fixed k. Math. Oper. Res. **19**(1), 24–37 (1994)
11. Guiñez, F., Queyranne, M.: The size-constrained submodular k-partition problem, Manuscript, https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxmbGF2aW9ndWluZXpob21lcGFnZXxneDo0NDVlMThkMDg4ZWRlOGI1 (2012)
12. Gupta, A., Lee, E., Li, J.: An FPT algorithm beating 2-approximation for k-cut. In: Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 2821–2837 (2018)
13. Gupta, Anupam, Lee, Euiwoong, Li, Jason.: Faster exact and approximate algorithms for k-cut. In: Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS '18, pp. 113–123 (2018)
14. Hardy, G., Littlewood, J., Pólya, G.: Inequalities, 2nd edn. Cambridge University Press, Cambridge (1952)
15. Kamidoi, Y., Wakabayashi, S., Yoshida, N.: A divide-and-conquer approach to the minimum k-way cut problem. Algorithmica **32**(2), 262–276 (2002)
16. Kamidoi, Y., Yoshida, N., Nagamochi, H.: A deterministic algorithm for finding all minimum k-way cuts. SIAM J. Comput. **36**(5), 1329–1341 (2007)
17. Karger, D., Stein, C.: A new approach to the minimum cut problem. J. ACM **43**(4), 601–640 (1996)
18. Klimmek, R., Wagner, F.: A simple hypergraph min cut algorithm. Internal Report B 96-02 Bericht FU Berlin Fachbereich Mathematik und Informatik (1995)
19. Kogan, D., Krauthgamer, R.: Sketching cuts in graphs and hypergraphs. In: Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS'15, pp. 367–376 (2015)
20. Lawler, E.: Cutsets and Partitions of Hypergraphs. Networks **3**, 275–285 (1973)
21. Mak, W.-K., Wong, D.: A fast hypergraph min-cut algorithm for circuit partitioning. Integr. VLSI J. **30**(1), 1–11 (2000)
22. Manurangsi, P.: Almost-polynomial ratio ETH-hardness of approximating densest k-subgraph. In: Proceedings of the 49th Annual ACM Symposium on Theory of Computing, STOC'17, pp. 954–961 (2017)
23. Manurangsi, P.: Inapproximability of maximum Biclique problems, minimum k-cut and densest at-least-k-subgraph from the small set expansion hypothesis. In: Proceedings of the 44th International Colloquium on Automata, Languages, and Programming, ICALP'17, pp. 79:1–79:14 (2017)
24. Okumoto, K., Fukunaga, T., Nagamochi, H.: Divide-and-conquer algorithms for partitioning hypergraphs and submodular systems. Algorithmica **62**(3), 787–806 (2012)
25. Raghavendra, P., Steurer, D.: Graph expansion and the unique games conjecture. In: Proceedings of the 42nd Annual ACM Symposium on Theory of Computing, STOC'10, pp. 755–764 (2010)
26. Saran, H., Vazirani, V.: Finding k cuts within twice the optimal. SIAM J. Comput. **24**(1), 101–108 (1995)
27. Thorup, M.: Minimum k-way cuts via deterministic greedy tree packing. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC'08, pp. 159–166 (2008)

28. Xiao, M.: An improved divide-and-conquer algorithm for finding all minimum k-way cuts. In: Proceedings of 19th International Symposium on Algorithms and Computation, ISAAC'08, pp. 208–219 (2008)
29. Xiao, M.: Finding minimum 3-way cuts in hypergraphs. Inf. Process. Lett. (Preliminary version in TAMC 2008) **110**(14), 554–558 (2010)
30. Zhang, P., Cai, J.-Y., Tang, L.-Q., Zhao, W.-B.: Approximation and hardness results for label cut and related problems. J. Comb. Optim. **21**(2), 192–208 (2011)
31. Zhang, P., Fu, B.: The label cut problem with respect to path length and label frequency. Theor. Comput. Sci. **648**, 72–83 (2016)
32. Zhao, L.: Approximation algorithms for partition and design problems in networks. Ph.D. thesisGraduate School of Informatics, Kyoto University, Japan (2002)
33. Zhao, L., Nagamochi, H., Ibaraki, T.: Greedy splitting algorithms for approximating multiway partition problems. Math. Program. **102**(1), 167–183 (2005)