Mimicking Networks for Constrained Multicuts in Hypergraphs

Kyungjin Cho
Department of Computer Science and Engineering, POSTECH, Pohang, Republic of Korea

Eunjin Oh
Department of Computer Science and Engineering, POSTECH, Pohang, Republic of Korea

Abstract

In this paper, we study a *multicut-mimicking network* for a hypergraph over terminals T with a parameter c. It is a hypergraph preserving the minimum multicut values of any set of pairs over T where the value is at most c. This is a new variant of the multicut-mimicking network of a graph in [Wahlström ICALP'20], which introduces a parameter c and extends it to handle hypergraphs. Additionally, it is a natural extension of the *connectivity-c mimicking network* introduced by [Chalermsook et al. SODA'21] and [Jiang et al. ESA'22] that is a (hyper)graph preserving the minimum cut values between two subsets of terminals where the value is at most c.

We propose an algorithm for a hypergraph that returns a multicut-mimicking network over terminals T with a parameter c having $|T|c^{O(r \log c)}$ hyperedges in $p^{1+o(1)} + |T|(c^r \log n)^{\tilde{O}(rc)} m$ time, where p and r are the total size and the rank, respectively, of the hypergraph.

2012 ACM Subject Classification Theory of computation → Sparsification and spanners

Keywords and phrases hyperedge multicut, vertex sparsification, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2024.21

Related Version *Full Version*: https://arxiv.org/abs/2409.12548

Funding This work was partly supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2024-00440239, Sublinear Scalable Algorithms for Large-Scale Data Analysis) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.RS-2024-00358505).
Kyungjin Cho: Supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.RS-2024-00410835).

1 Introduction

Graph sparsification is a fundamental tool in theoretical computer science. By reducing the size of a graph while preserving specific properties, such as the value of an objective function or its approximation, graph sparsification significantly enhances computational efficiency. This is particularly crucial for practical applications with limited resources and for handling large-scale data in real-world problems. Due to these advantages, various types of sparsification results have been presented over the decades, including spanners [4, 9], flow sparsification [6, 15], and cut sparsification [5]. Additionally, their applications have been widely studied, such as in designing dynamic algorithms [11]. In this paper, we focus on graph sparsification specifically tailored for hypergraph separation and cut problems.

Hypergraph separation and cut problems have garnered significant attention due to their extensive applications and theoretical challenges. These problems are particularly compelling because hypergraphs offer more accurate modeling of many complex real-world scenarios compared to normal graphs. Examples include VLSI layout [1], data-pattern-based clustering [23], and social tagging networks [25]. The transition from graph to hypergraph separation problems opens up new avenues for research, driven by the need to address the unique properties and complexities inherent in hypergraphs. Researchers have thus increasingly focused on developing graph algorithms and theoretical frameworks for hypergraph problems, such as the small set expansion problem in hypergraphs [19], spectral sparsification in hypergraphs [2, 13], and connectivity-c mimicking problem in hypergraphs [10]. This growing interest underscores the critical importance of hypergraph separation and cut problems in both theoretical and practical applications.

One of the key problems in (hyper)graph sparsification is the *mimicking problem.* It aims to find a graph that preserves minimum cut sizes between any two subsets of vertices called *terminals.* A *cut* between two sets of vertices is a set of edges whose removal disconnects the given two sets. Kratsch et al. [14] showed that there is a mimicking network with $O(\tau^3)$ edges, where τ is the number of edges incident to terminals. Chalermsook et al. [3] introduced a *constraint version*, called *connectivity-c mimicking problem*, that aims to preserve minimum cut sizes between every two subsets of terminals where the size is at most c, and they showed that there is such a graph with $O(kc^4)$ edges, which was later improved to $O(kc^3)$ [16], where k is the number of terminals. This result was extended to hypergraphs by Jiang et al. [10].

A crucial variant of the mimicking problem is the *multicut-mimicking problem.* A *multicut* of pairs of vertices is a set of (hyper)edges whose removal disconnects each given pair. Studies have shown that the multicut problem is highly beneficial in various applications, including network design, optimization, and security, where maintaining specific connectivity while minimizing resources is necessary compared to cut problems [12]. It is already known that the problem is NP-hard even for graphs [8, 20]. A *multicut-mimicking network* for a set of terminals in a (hyper)graph is a (hyper)graph that preserves the size of the minimum multicut for any set of pairs of terminals. Kratsch et al. [14] proposed a method to obtain a multicut-mimicking network by contracting edges in a graph except at most $\tau^{O(k)}$ edges, where k and τ are the numbers of terminals and incident edges to terminals, respectively. Wahlström [24] refined this method and reduced the number of edges to $\tau^{O(\log \tau)}$.

Unlike the mimicking problem, there is no existing result for the constraint version of *multicut-mimicking network problem*, even for graphs. We further study the multicut-mimicking problem by introducing a parameter c. Precisely, we present an algorithm to compute a hypergraph, that preserves the size of the minimum multicut for any set of pairs of terminals where the size is at most c, with a linear size in the number of terminals while the previous best-known result for multicut-mimicking network, without the parameter c, has an exponential size [14]. It will allow for more refined control over the sparsification process, enabling the construction of smaller and more efficient networks. For instance, this notion in mimicking problem was utilized for a dynamic connectivity problem [11].

Our result. In this paper, we study *vertex sparsifiers for multiway connectivity* with a parameter $c > 0$. Our instance (G, T, c) consists of an undirected hypergraph G, terminal set $T \subseteq V(G)$, and a parameter c. Precisely, we construct a hypergraph that preserves minimum multicut values over T where the value is at most c. It is the first result for the multicut-mimicking networks adapting the parameter c even for graphs.

Previously, the best-known multicut-mimicking network had a quasipolynomial size in the total degree of terminals in T [24], specifically $|\partial T|^{O(\log |\partial T|)}$. By introducing the parameter c, we demonstrate that a multicut-mimicking network for (G, T, c) exists with a size linear in $|T|$. This allows us to utilize the near-linear time framework of Jiang et al. [10] to find a mimicking network using the *expander decomposition* of Long and Saranurak [18]. Our result is summarized in Theorem 1. Here, $m = |E(G)|$ and r is the rank of G.

▶ **Theorem 1.** *For (G, T, c), we can compute a multicut-mimicking network of at most $kc^{O(r \log c)}$ hyperedges in $p^{1+o(1)} + k(c^{r \log c} \log n)^{O(rc)} m$ time, where $k = |T|$ and $p = \sum_{e \in E} |e|$.*

Outline. Our work extends the framework from connectivity-c mimicking networks for hypergraphs, introduced by Jiang et al. [10], to multicut-mimicking networks, as well as adapting methods from multicut-mimicking networks for graphs, introduced by Wahlström [24], to hypergraphs with a parameter c. While we broadly follow the previous approaches, we extend the concepts and methods used in the previous studies to fit the multicut-mimicking problem in hypergraphs with the parameter c. This extension allows us to handle the complexities of hypergraphs effectively.

We introduce notions used in this paper in Section 2 and illustrate an efficient algorithm to compute a small-sized multicut-mimicking network outlined in Theorem 1 in Section 3. We give an upper bound for the size of minimal multicut-mimicking networks of hypergraphs in Section 4, which is a witness for the performances of our algorithm outlined in Theorem 1.

2 Preliminaries

A *hypergraph* G is a pair $(V(G), E(G))$, where $V(G)$ denotes the set of vertices and $E(G)$ is a collection of subsets of $V(G)$ referred to as *hyperedges*. If the context is clear, we write V and E. The *rank* of G is defined as the size of its largest hyperedge. For a vertex $v \in V$, a hyperedge e is said to be *incident* to v if $v \in e$. For a vertex set $X \subset V$, let $\partial_G X$ denote the set of hyperedges in E containing at least one vertex from X and one from $V \setminus X$, and let $E(X)$ denote the set of hyperedges fully contained in X. Additionally, we let G/e denote the *contraction* of a hyperedge e in G obtained by merging all vertices in e into a single vertex and modifying the other hyperedges accordingly. A *path* in a hypergraph is defined as a sequence of hyperedges such that any two consecutive hyperedges contain a common vertex.

Consider a *partition* $(X_1, \ldots, X_s)$ of a vertex set $X \subseteq V$. We call each subset X_i a *component* of this partition. Additionally, the *cut* of $(X_1, \ldots, X_s)$ in G is defined as the set of hyperedges of G intersecting two different components. We let $[a] = \{1, \ldots, a\}$ and $[a, b] = \{a, \ldots, b\}$ for integers $a < b$. Furthermore, let $|X|$ denote the number of elements of a set X. For a hyperedge $e \in E(G)$, we let $|e|$ to denote the number of vertices in e.

In this paper, an instance (G, T, c) consists of a hypergraph G, a set $T \subseteq V(G)$, and a positive constant c. We refer to the vertices in T as *terminals*. For a set R of pairs of T, a *multicut* of R in G is a set of hyperedges $F \subset E(G)$ such that every connected component in $G \setminus F$ contains at most one element of every pair $\{t, t'\} \in R$. We construct a *multicut-mimicking network* H of (G, T, c) that is a hypergraph obtained from G by contraction of hyperedges which preserves the size of minimum multicut for all set R of pairs over T where the size is at most c. Precisely, if a multicut F of R exists in G with $|F| \leq c$, then a multicut F' of R exists in H with $|F'| \leq |F|$. We say H is *minimal* if no contraction H/e is a multicut-mimicking network of (G, T, c). Analogously, we define a *minimal instance*. We address the multicut-mimicking problem by utilizing *multiway cuts*.

2.1 Multiway Cuts and Essential Edges

We refer to a partition of terminals T as a *terminal partition*. For a terminal partition $\mathcal{T}$, a hyperedge set F is termed *a multiway cut of* $\mathcal{T}$ if any two terminals from different components in $\mathcal{T}$ are not connected in $G \setminus F$. Furthermore, if there is no multiway cut of size less than $|F|$, then F is called a *minimum multiway cut of* $\mathcal{T}$ *in* G. Let $\mathsf{min\text{-}cut}_G(\mathcal{T})$

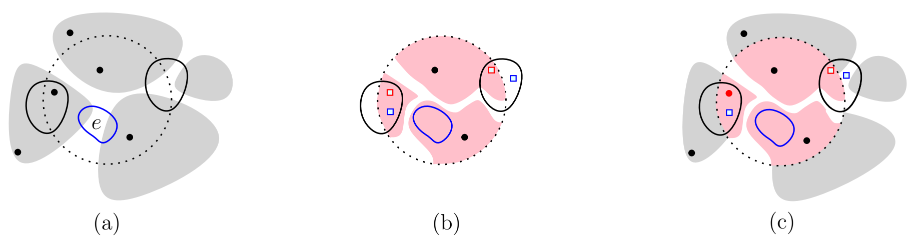

Figure 1 Sketch of Lemma 3. For a multiway cut F of a terminal partition $\mathcal{T}$ and $X \subset V$, illustrated in (a), let $\mathcal{T}'$ be the terminal partition of T_X in $(\hat{G}[X], T_X, c_X)$ according to $G \setminus F$. Then we can modify F as excluding e if e is non-essential in $(\hat{G}[X], T_X, c_X)$, illustrated in (b-c).

denote the minimum multiway cut size of the partition in G. A hyperedge $e \in E(G)$ is said to be *essential* for (G, T, c) if there exists a terminal partition $\mathcal{T}$ with $\text{min-cut}_G(\mathcal{T}) \leq c$ such that every minimum multiway cut of $\mathcal{T}$ in G contains e. Otherwise, e is *non-essential*.

Multicuts and multiway cuts in graphs are closely related [24, Proposition 2.2]. We observe that this close relation also holds in hypergraphs. Briefly, a contraction of a non-essential hyperedge is a multicut-mimicking network by Lemma 2, proved in the full version.

▶ **Lemma 2.** *For a hyperedge e of G, G/e is a multicut-mimicking network for (G, T, c) if and only if e is non-essential for (G, T, c).*

A multicut-mimicking network is minimal if and only if every hyperedge is essential. Note that we cannot contract multiple non-essential hyperedges simultaneously. This is because even if two hyperedges e and e' are non-essential in (G, T, c), the contraction $G/\{e, e'\}$ might not be a multicut-mimicking network of (G, T, c) even for a graph G, not a hypergraph. In this paper, we construct a multicut-mimicking network by finding and contracting non-essential hyperedges one by one.

2.2 Restricted Hypergraphs and Subinstances

The *subinstance* $(\hat{G}[X], T_X, c_X)$ of (G, T, c) for $X \subset V(G)$ is constructed as follows. Refer to Figure 1 (a-b). For each hyperedge $e \in \partial X$, we insert a vertex a_e, and we choose an arbitrary terminal t_e in $e \cap (X \cap T)$. If no such terminal exists, we insert a new vertex t_e. We refer to a_e, t_e as the *anchored terminals* of e, and $(e \cap X) \cup \{a_e, t_e\}$ as the *restricted hyperedge* of e, denoted by e_X. We obtain $\hat{G}[X]$ from G through the following: i) add the anchored terminals of the hyperedges in ∂X, ii) replace the hyperedges in ∂X with their restricted hyperedges, and iii) delete the vertices $V \setminus X$ and the hyperedges $E(V \setminus X)$. We call it the *restricted hypergraph* of G for X. Let T_X denote the set of all terminals in $T \cap X$ and the anchored terminals, and $c_X = \min\{c, |T_X|\}$. All subinstances preserve all essential hyperedges in the original instance by Lemma 3. Figure 1 sketches its proof, and details are in the full version.

▶ **Lemma 3.** *If a hyperedge e is non-essential in a subinstance $(\hat{G}[X], T_X, c_X)$, then e is also non-essential in the original instance (G, T, c). Furthermore, e is not in $\partial_G X$.*

3 Efficient Algorithm for Computing Multicut-Mimicking Networks

In this section, we design an algorithm to compute a minimal multicut-mimicking network for (G, T, c), where G is a hypergraph. We broadly follow the approach of Jiang et al. [10]. Since their original algorithm was designed for mimicking networks, not multicut-mimicking networks, we need to modify their algorithm. First, we introduce their algorithm briefly.

Jiang et al. [10] designed an algorithm to find a connectivity-c mimicking network for hypergraphs using the *expander decomposition* of Long and Saranurak [18]. Precisely, they designed an algorithm to find a connectivity-c mimicking network of size linear in $|T|$ for an *expander* G with terminals T, and then, they extended it for a general hypergraph using the expander decomposition. For a parameter $\phi > 0$, a hypergraph G (and instance (G, T, c)) is called a ϕ-*expander* if either $E(X)$ or $E(V \setminus X)$ has at most $\phi^{-1}|\partial X|$ hyperedges for any vertex set $X \subset V(G)$. The following explains the key idea of their and our algorithm.

Recall that contracting a non-essential hyperedge obtains a smaller mimicking network by Lemma 2. Generally, a non-essential hyperedge can be found by comparing every terminal partition and subset of hyperedges, which is time-consuming. However, if we suppose that the given instance is an *expander*, then we can do this more efficiently by comparing *useful terminal partitions* and their minimum multiway cuts instead of whole partitions and hyperedge subsets. We adapt concepts used in the previous research such as *useful terminal partitions* to suit our needs, and we newly introduce the concept *core* of a multiway cut which refers to a small-sized vertex set including whole hyperedges of the multiway cut.

Useful terminal partitions, connected multiway cuts, and cores. Assume that the instance (G, T, c) is a ϕ-expander and G is a connected hypergraph. For a multiway cut F in G, we define the *core* of F as the union C of connected components X in $G \setminus F$ with $|E(X)| \leq \phi^{-1}|F|$. The definition of an expander guarantees that at most one component in $G \setminus F$ has more than $\phi^{-1}|F|$ hyperedges. Therefore, the multiway cut F includes all hyperedges ∂C and is contained in $E(C) \cup \partial C$. We say F is a *connected multiway cut* in (G, T, c) if it is a minimum multiway cut of some terminal partition with $|F| \leq c$ and $T \cap C$ is connected in $\hat{G}[C]$, where C is the core of F and $\hat{G}[C]$ is the restricted hypergraph defined in Section 2.2. A terminal partition is said *useful* if every minimum multiway cut of it is a connected multiway cut.

Since every core of connected multiway cuts has a small number of vertices and hyperedges in ϕ-expander, we can enumerate all of them efficiently. Additionally, since a core includes its corresponding multiway cut, we can also enumerate all connected multiway cuts and useful partitions. Details are in Section 3.1. The most interesting property is that comparing all useful partitions is sufficient to find a non-essential hyperedge.

▶ **Lemma 4.** *A hyperedge $e \in E$ is essential for (G, T, c) if and only if there is a useful partition such that every minimum multiway cut for it contains e.*

Proof. The "if" direction is trivial since it is consistent with the definition of essential. For the "only if" direction, we assume that e is essential for (G, T, c). Let $\mathcal{T}$ be a terminal partition minimizing $\textsf{min-cut}_G(\mathcal{T})$ of which every minimum multiway cut involves e. In the following, we show that $\mathcal{T}$ is a *useful partition* by contradiction. Figure 2 illustrates this proof.

Assume that $\mathcal{T}$ is not a useful partition for (G, T, c), and let F be a minimum multiway cut of $\mathcal{T}$ which is not a connected multiway cut. If the core of F is $V(G)$, then F is a connected multiway cut. Therefore, the core is the complement of some connected component X in $G \setminus F$. Let C be the connected component in $G - X$ that intersects the hyperedge e. Refer to Figure 2(a). Then we decompose the multiway cut F into F_e and $\bar{F}$ where $F_e = F \cap E(C \cup X)$ and $\bar{F} = F \setminus F_e$. By the construction, we have $F_e \subsetneq F$ and $e \in F_e$. We construct a minimum multiway cut of $\mathcal{T}$ excluding e that completes the proof.

Let $\mathcal{T}'$ be the terminal partition according to $G \setminus F_e$. Since $F_e \subsetneq F$ and we chose $\mathcal{T}$ as minimizing $\textsf{min-cut}_G(\mathcal{T})$ while any minimum multiway cut of $\mathcal{T}$ contains e, there is a multiway cut F' of $\mathcal{T}'$ excluding e. We claim that $F' \cup \bar{F}$ is a minimum multiway cut of $\mathcal{T}$

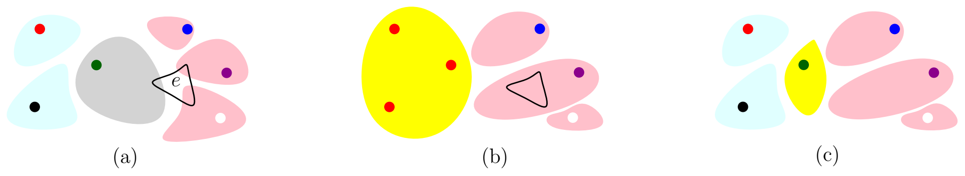

Figure 2 Illustration of the proof of Lemma 4. (a) Illustration of the terminal partition $\mathcal{T}$ and the vertex partition according to $G \setminus F$. The middle gray area is X. The right three red areas form C. (b) Illustration of the terminals partition $\mathcal{T}'$ and the partition of $G \setminus F'$. (c) Illustration of the partition $G \setminus (F' \cup \bar{F})$. $F' \cup \bar{F}$ is a multiway cut of $\mathcal{T}$ excluding e.

excluding e which contradicts and completes the proof. Refer to Figure 2(b-c). Note that the multiway cut has a size at most $|F|$ and excludes e by construction. Thus, it is sufficient to show that there is no path in $G \setminus (F' \cup \bar{F})$ between two components in $\mathcal{T}$.

Consider a path π in G between two terminals in different components in $\mathcal{T}$. Note that π is not a path in $G \setminus F$, and thus, π is not in $G \setminus F_e$ or $G \setminus \bar{F}$. Recall that F' is the multiway cut of the terminal partition according to $G \setminus F_e$. That means π is not in $G \setminus F'$ if it is not in $G \setminus F_e$. Therefore, there is no path in $G \setminus (F' \cup \bar{F})$ between two components in $\mathcal{T}$. ◀

3.1 Useful Terminal Partitions in Expanders

In this section, we explain how to efficiently enumerate all useful terminal partitions and their minimum multiway cuts. Then, we explain how to compute a multicut-mimicking network for an expander using the enumerated list along with Lemma 4. Here, the instance (G, T, c) is a ϕ-expander and G is a connected hypergraph.

The key is based on Observation 5, proved in the full version. Briefly, cores in a ϕ-expander have a small number of vertices and hyperedges. The observation enables us to find all cores of connected multiway cuts and subsequently enumerate all useful partitions. However, it is possible to enumerate terminal partitions that are not useful. Therefore, we need to *prune* the enumerated lists for useful terminal partitions and their minimum multiway cuts.

▶ **Observation 5.** *For a connected multiway cut F and its core C, the restricted hypergraph $\hat{G}[C]$ is connected and has at most $(3\phi^{-1}+1)|F|$ hyperedges.*

Enumerating connected multiway cuts. Jiang et al. [10, EnumerateCutsHelp] designed an algorithm to enumerate all connected vertex sets C with $|\partial C| \leq c$, $|E(C)| \leq M$, and $t \in C$ in the ϕ-expander G when a vertex $t \in V(G)$ and two integers c, M are given as an input. This algorithm takes $(r(M+c))^{O(rc)}$ time. Additionally, the number of returned connected vertex sets is at most $(r(M+c))^{O(rc)}$. We use this algorithm along with Observation 5 for enumerating all connected multiway cuts. Details are in the full version. Briefly, we find all connected vertex set C with $C \cap T \neq \emptyset$, $|\partial C| \leq c$, and $|E(C)| \leq (3\phi^{-1}+1)c$. Then we can enumerate every connected multiway cut from c sized subsets of $E(C) \cup \partial C$ by Observation 5.

In conclusion, we enumerate $|T|(rc\phi^{-1})^{O(rc)}$ multiway cuts including all connected multiway cuts of size at most c in $|T|(rc\phi^{-1})^{O(rc)}$ time. At most $|T|(rc\phi^{-1})^{O(rc)}$ terminal partitions of T are contributed by the enumerated multiway cuts since each multiway cut of size c generates at most rc connected components. They include all useful partitions.

Pruning useful terminal partitions. To check whether a terminal partition is useful or not, we need to verify if $C \cap T$ is connected in $\hat{G}[C]$ for each core C of its minimum multiway cuts. Specifically, it is sufficient to check the inclusion-wise minimal cores among them. For this, we utilize *important cuts* ∂R, which inclusion-wise maximizes $R \subset V$ while maintaining the size of the cut ∂R. The definition aligns with our needs as outlined in Lemma 6, proved in the full version.

For two disjoint vertex sets A and B, let R be a vertex set containing A while excluding B. We say ∂R is an *important cut* of (A, B) if there is no $R' \supsetneq R$ excluding B with $|\partial R'| \leq |\partial R|$. This definition holds even if $A = \emptyset$. In a directed graph, an important cut is defined analogously by setting ∂R as the outgoing arcs from R to $V \setminus R$. Furthermore, there is an FPT algorithm for enumerating all important cuts in a directed graph [7].

▶ **Lemma 6.** *For a terminal partition $\mathcal{T}$ with $\mathsf{min\text{-}cut}_G(\mathcal{T}) \leq c$, the following are equivalent:*

(i) *$\mathcal{T}$ is a useful terminal partition of T, and*

(ii) *Every minimum multiway cut F of $\mathcal{T}$ is a connected multiway cut if some component T' in $\mathcal{T}$ and important cut R of $(T', T \setminus T')$ satisfy $\partial R \subset F$, $|E(R)| \geq c\phi^{-1}$, and $F \cap E(R) = \emptyset$.*

We construct an auxiliary directed graph D^{inc} to enumerate important cuts in G. For instance (G, T, c), the vertex set of D^{inc} is the union of $V(G)$ and two copies E_{in} and E_{out} of $E(G)$. For a hyperedge e in $E(G)$, we use e_{in} and e_{out} to denote the copies of e in E_{in} and E_{out}, respectively, and we insert one arc from e_{in} to e_{out}. For each $v \in V(G)$ and $e \in E(G)$ with $v \in e$, we insert $2c$ parallel arcs from v to e_{in} and from e_{out} to v. Important cuts in G correspond one-to-one with those in D^{inc}. Details are in the full version.

We can enumerate all important cuts of G by applying the FPT algorithm for D^{inc}. There are at most $2^{O(rc)}$ number of important cuts in G, and they can be enumerated in $2^{O(rc)}m$ time [7], where $m = |E(G)|$. By Lemma 6, we can prune all useful terminal partitions in $|T|(rc\phi^{-1})^{O(rc)}m$ time among the $|T|(rc\phi^{-1})^{O(rc)}$ enumerated candidates.

Lemma 7 summarizes this section, details are in the full version. In the remainder, we give an algorithm to compute a minimal multicut-mimicking network for ϕ-expander using it.

▶ **Lemma 7.** *There are $|T|(rc\phi^{-1})^{O(rc)}$ useful partitions and their minimum multiway cuts in a ϕ-expander (G, T, c). We can enumerate all of them in $|T|(rc\phi^{-1})^{O(rc)}m$ time.*

Algorithm for ϕ-expanders. By Lemma 4 and Lemma 7, we can recursively find and contract a non-essential hyperedge efficiently for a ϕ-expander (G, T, c) until every hyperedge in the instance is essential. The algorithm is outlined in the following lemma. Recall that a *minimal multicut-mimicking network* H of (G, T, c) is a multicut-mimicking network so that every hyperedge in (H, T, c) is essential. Here, $m = |E(G)|$ and r is the rank of G.

▶ **Lemma 8.** *For a ϕ-expander (G, T, c), we can find a minimal multicut-mimicking network in $|T|(rc\phi^{-1})^{O(rc)}m$ time. Moreover, it has at most $|T|c^{O(r \log c)}$ hyperedges.*

Sketch of the proof. We demonstrate that a minimal multicut-mimicking network has at most $|T|c^{O(r \log c)}$ hyperedges in Section 4 which is one of our main contributions. We sketch an algorithm to compute a minimal multicut-mimicking network, details are in the full version.

If $m \in O(c\phi^{-1})$, then we can obtain such a multicut-mimicking network by enumerating all multiway cuts of size at most c. In the following, we consider the other case that $m \geq 3c\phi^{-1} + c$. At the beginning of the algorithm, we enumerate all important cuts and their minimum multiway cuts by applying Lemma 7. Since there are at most $k(rc\phi^{-1})^{O(rc)}$ multiway cuts and each multiway cut consists at most $(rc)^{O(rc)}$ useful partitions, we visit a hyperedge and check whether it is non-essential in the current instance in $k(rc\phi^{-1})^{O(rc)}$ time by Lemma 4. If it is non-essential, we contract it before we move to another hyperedge.

After we contract a non-essential hyperedge, we do not need to call the algorithm outlined in Lemma 7 again. Precisely, deleting multiway cuts which include the contracted hyperedge among the already enumerated ones is sufficient while more than $(3c\phi^{-1} + c)$ hyperedges are left. This is because no new connected multiway cut occurs or disappears by contracting a non-essential hyperedge if the number of remaining hyperedges exceeds $(3c\phi^{-1} + c)$. Its detailed proof is in the full version. If all remaining hyperedges are essential, then we return the current instance as a solution. For the other case that the number of them is at most $(3c\phi^{-1} + c)$, we apply the algorithm for $m \in O(c\phi^{-1})$ explained before. In conclusion, we can obtain a minimal multicut-mimicking network in the time complexity in Lemma 8. ◀

3.2 Near-Linear Time Algorithm for General Hypergraphs

In this section, we obtain a multicut-mimicking network for a general instance (G, T, c), by recursively calling MimickingExpander. The submodule MimickingExpander computes a small multicut-mimicking network based on the *expander decomposition* of Long and Saranurak [18] and the algorithm for a ϕ-expander with $\phi > 0$ outlined in Lemma 8. However, its return is not sufficiently small, and thus, we obtain a much smaller solution by applying it recursively.

MimickingExpander$(G; T, c)$. We let $n = |V(G)|$, $m = |E(G)|$, and $\phi^{-1} = 4rc^{Mr\log c}\log^3 n$, where the multicut-mimicking network returned by Lemma 8 has at most $kc^{Mr\log c}$ hyperedges for an expander with k terminals. When the submodule is called, we first decompose the vertex set $V(G)$ into the vertex partition $(V_1, \ldots, V_s)$ so that the size of the cut of $(V_1, \ldots, V_s)$ is at most $\phi m \log^3 n$ and each $\hat{G}[V_j]$ is a ϕ-expander for $j \in [s]$. There is a $p^{1+o(1)}$ time algorithm computing such a vertex partition [18], where $p = \sum_{e\in E} |e|$.

Every subinstance $(\hat{G}[V_j], T_j, c_j)$ is a ϕ-expander, where $c_j = \min\{c, |T_j|\}$ and T_j is the union of $T \cap V_j$ and anchored terminals in $\hat{G}[V_j]$. For each $(\hat{G}[V_j], T_j, c_j)$, we construct a multicut-mimicking network H_j by Lemma 8. Finally, we return the multicut-mimicking network H by gluing $H_1, \ldots, H_s$. Precisely, we merge all restricted hyperedges having a common anchored terminal and remove all anchored terminals not in V. The following lemma summarizes the performance of this submodule, proved in the full version.

▶ **Lemma 9.** MimickingExpander$(G; T, c)$ *returns a multicut-mimicking network of* (G, T, c) *with* $(|T|c^{O(r\log c)} + (m/2))$ *hyperedges in* $(p^{1+o(1)} + |T|(c^{r\log c}\log n)^{O(rc)}m)$ *time.*

Overall algorithm. For an instance (G, T, c), we initialize $G_0 = G$ and obtain the multicut-mimicking network G_i by MimickingExpander$(G_{i-1}; T, c)$ for $i \in [\lceil \log m \rceil]$, inductively. Finally, we return $G_{\lceil \log m \rceil}$. This algorithm corresponds to Theorem 1. Details are in the full version.

▶ **Theorem 1.** *For* (G, T, c)*, we can compute a multicut-mimicking network of at most* $kc^{O(r\log c)}$ *hyperedges in* $p^{1+o(1)} + k(c^{r\log c}\log n)^{O(rc)}m$ *time, where* $k = |T|$ *and* $p = \sum_{e\in E} |e|$.

4 Bound for Minimal Instances

To complete the proof of Lemma 8 (and Theorem 1), we need to show that a minimal multicut-mimicking network for an expander (G, T, c) has at most $|T|c^{O(r\log c)}$ hyperedges. This section demonstrates it for not only expanders but also general instances. Precisely, we show the following theorem in this section. Here, r is the rank of G.

▶ **Theorem 10.** *Every minimal instance* (G, T, c) *has at most* $|T|c^{O(r\log c)}$ *hyperedges.*

In this section, we consider the scenario that every terminal in T has degree one in G. It is sufficient since the other case can be reduced to this scenario by inserting $c+1$ dummy terminals instead of each terminal t in T that are adjacent only to t. This reduction does not increase the rank or the parameter c while increasing the number of terminals by at most c times. However, this increase does not affect the asymptotic complexity in Theorem 10. The following explains the previous works and introduces the notions used in this section.

We broadly follow the approach of Wahlström [24]. He utilized the framework of Kratsch et al. [14] for multicut-mimicking networks in graphs without the parameter c. We incorporate the *unbreakable* concept to address the parameter c. Additionally, we slightly modify the *dense* concept used in the previous work to account for hypergraphs.

Unbreakable and dense. For a subinstance $(\hat{G}[X], T_X, c_X)$ with respect to $X \subset V(G)$, the terminal set T_X includes $T \cap X$ and up to two anchored terminals for each restricted hyperedge $e \in \partial_G X$. Hence, $|T_X| \leq 2|\partial_G X| + |T \cap X|$. We denote the value $2|\partial_G X| + |T \cap X|$ as $\mathsf{cap}_T(X; G)$, or $\mathsf{cap}_T(X)$ if the context is clear. Furthermore, we define the following:

- **Unbreakable:** An instance (G, T, c) is said to be d-unbreakable for $d > 0$ if $|T \cap X| \leq d$ for any vertex set $X \subseteq V(G)$ with $|T \cap X| \leq |T \setminus X|$ and $|\partial X| \leq c$.
- **Dense:** An instance (G, T, c) is said to be α-dense for $\alpha > 0$ if $|E(X)| \leq (\mathsf{cap}_T(X))^\alpha$ for any vertex set $X \subseteq V$ with $0 < |E(X)| \leq |E(V \setminus X)|$ and $|\partial X| \leq c$.

Wahlström [24] also used the concept of *dense* defining it based on the vertices instead of $E(\cdot)$. Since he addressed graphs, it was guaranteed that $|E(X)| = \Omega(|X|)$ by using connective assumption. However, this is not for hypergraphs. Thus, we slightly modify the definition.

In Section 4.2, we prove Theorem 10 for *unbreakable and dense* instances using the notions introduced in Section 4.1. Section 4.3 explains how to extend this proof for general instances.

4.1 Matroids and Representative Sets

We use the notion of matroids and representative sets as in the previous work, which is a generalization of the notion of linear independence in vector spaces. Formally, a matroid $(S, \mathcal{I})$ consists of a *universe set* S and an *independent set* $\mathcal{I} \subseteq 2^S$ with $\emptyset \in \mathcal{I}$ satisfying:

- If $B \in \mathcal{I}$ and $A \subseteq B$, then $A \in \mathcal{I}$, and
- If $A, B \in \mathcal{I}$ with $|A| < |B|$, then there exists $x \in B \setminus A$ such that $A \cup \{x\} \in \mathcal{I}$.

For a matroid $(S, \mathcal{I})$, its *rank* is the size of the largest set in $\mathcal{I}$. It is said to be *representable* if there is a matrix over a field whose columns are indexed by the elements of S such that: $F \subset S$ is in $\mathcal{I}$ if and only if the columns indexed by F are linearly independent over the field.

Representative sets. Kratsch et al. [14] introduced a framework for computing non-essential vertices using the notion of *representative sets*. We employ the framework. For this purpose, we introduce two operations: *truncation* and *direct sum* along with Lemma 11. For a matroid $(S, \mathcal{I})$ and an integer $r > 0$, the r-*truncation* of $(S, \mathcal{I})$ is defined as a matroid $(S, \mathcal{I}')$ such that $F \subseteq S$ is contained in $\mathcal{I}'$ if and only if $|F| \leq r$ and $F \in \mathcal{I}$. Note that an r-truncation has rank at most r. For matroids $\mathcal{M}_1, \ldots, \mathcal{M}_s$ over disjoint universes with $\mathcal{M}_i = (S_i, \mathcal{I}_i)$ for $i \in [s]$, their *direct sum* is defined as a matroid $(S, \mathcal{I})$ such that S is the union of all S_i, and a subset F of S is in $\mathcal{I}$ if and only if F can be decomposed into s disjoint subsets, each of which is independent in $\mathcal{M}_i$. The direct sum of matroids also satisfies the matroid axioms.

For a matroid $(S, \mathcal{I})$ and two subsets A, B of S, we say A *extends* B if $A \cap B = \emptyset$ and $A \cup B \in \mathcal{I}$. For $\mathcal{J} \subseteq 2^S$, a subset $\mathcal{J}^*$ of $\mathcal{J}$ is called a *representative set* if for any set $B \subseteq S$, there is a set $A^* \in \mathcal{J}^*$ that extends B when there is a set $A \in \mathcal{J}$ that extends B.

▶ **Lemma 11** ([14, Lemma 3.4]). *Let $\mathcal{M}_1 = (S_1, \mathcal{I}_1), \dots, \mathcal{M}_s = (S_s, \mathcal{I}_s)$ be matroids represented over the same finite field and with pairwise disjoint universe sets. Let $\mathcal{J}$ be a collection of sets containing one element from S_i for each $i \in [s]$. Then, some representative set of $\mathcal{J}$ in the direct sum of all matroids $\mathcal{M}_i$ has a size at most the product of the ranks of all $\mathcal{M}_i$.*

Uniform and (hyperedge) gammoids. We use two classes of representable matroids: *uniform matroids* and *gammoids*. They, along with their truncation and direct sum, are representable over any large field [7, 14, 17, 21, 22].

- **Uniform matroid:** For a set S and an integer $r > 0$, the *uniform matroid* on S of rank r is the matroid in which the universe set is S and the independent set consists of the sets containing at most r elements in S.
- **Gammoid:** For a directed graph $D = (V, A)$ and two subsets S and U of V, a *gammoid* defined on (D, S, U) is a matroid $(U, \mathcal{I})$ where $\mathcal{I}$ consists of the sets $X \subseteq U$ such that there are $|X|$ pairwise vertex-disjoint paths in D from S to X.

  In this paper, we define and use *hyperedge gammoids* to handle hypergraphs.

- **Hyperedge gammoid:** For a hypergraph G and a terminal set $T \subset V(G)$, we consider a directed graph D^{split} defined as follows. First, we start from the undirected graph D^{split} of which the vertex set is $E(G)$ and $ee' \in E(D^{\mathsf{split}})$ for two e, e' in $E(G)$ (and $V(D^{\mathsf{split}})$) if and only if $e \cap e'$ is not empty. Then, we replace each undirected edge with two-way directed arcs. Furthermore, we insert a copy E_{sink} of $E(G)$ into $V(D^{\mathsf{split}})$. We call the copy in E_{sink} of a hyperedge $e \in E(G)$ a *sink-only copy* of e, and denote it by $\mathsf{sink}(e)$. We insert one-way arcs from e' to $\mathsf{sink}(e)$ on D^{split} for every $e' \in E(G)$ where $e' \cap e$ is not empty. The *hyperedge gammoid* of (G, T) is the gammoid on $(D^{\mathsf{split}}, \partial_G T \subset E(G), E(G) \cup E_{\mathsf{sink}})$.

Recall that a *path* of a hypergraph is defined as a sequence of hyperedges such that any two consecutive hyperedges contain a common vertex. If a path starts or ends at a hyperedge containing a terminal t, we say that it starts or ends at t to make the description easier. For a subset F of $E(G) \cup E_{\mathsf{sink}}$, let F_E be the set of hyperedges e of $E(G)$ where e or $\mathsf{sink}(e)$ is contained in F. Even if F contains both e and $\mathsf{sink}(e)$, e appears in F_E exactly once.

▶ **Observation 12.** *$F \subseteq E(G) \cup E_{sink}$ is independent in the hyperedge gammoid of (G, T) if and only if there exist $|F|$ pairwise edge-disjoint paths from T to F_E in G with two exceptions:*
- *Two paths can end at $e \in E(G)$ if $e \in F$ and $\mathsf{sink}(e) \in F$, and*
- *One path can pass through e while another path ends at e if $e \notin F$ but $\mathsf{sink}(e) \in F$.*

4.2 Essential Hyperedges in Unbreakable and Dense Instances

Assume that (G, T, c) is d-unbreakable and α-dense with $c \le d \le |T|$ and $\alpha \ge 35r \log d$, where r is the rank of G. In this section, we show that (G, T, c) contains at most $|T|d^{\alpha-1}$ essential hyperedges which directly implies Theorem 10 for $O(c)$-unbreakable and $\Theta(r \log c)$-dense instances. Precisely, if there are more than $|T|d^{\alpha-1}$ hyperedges, then at least one is non-essential, and thus, the instance is not minimal. Here, $k = |T|$ and r is the rank of G.

The following matroids would be a witness for our claim with $i_0 = 30r$:
- One uniform matroid on E of rank $(\kappa + c)$, where $\kappa = (d/2)^{\alpha - i_0 - 2}$,
- One $(k + c + 1)$-truncation $\mathcal{M}_0$ of the hyperedge gammoid of (G, T), and
- i_0 copies $\mathcal{M}_1, \dots, \mathcal{M}_{i_0}$ of the $(d + c + 1)$-truncation of the hyperedge gammoid of (G, T),

For a hyperedge e, its appearance in the universe sets of the uniform matroid on E is denoted by e^{u}. In the universe set of $\mathcal{M}_i$ for $i \in [0, i_0]$, e_i and $\mathsf{sink}_i(e)$ denote the hyperedge and its sink-only copy, respectively. Note that sink-only copies have no outgoing arc in D^{split}, where the hyperedge gammoid is defined on the directed graph D^{split}, refer to Section 4.1.

Let $\mathcal{M}$ denote the direct sum of the $(i_0 + 2)$ matroids described above. For a hyperedge $e \in E(G)$, let $J(e) = \{e^{\mathsf{u}}\} \cup \{\mathsf{sink}_0(e), \ldots, \mathsf{sink}_{i_0}(e)\}$. Then, we let $\mathcal{J}^*$ be the representative set of $\{J(e) \mid e \in E(G)\}$ outlined in Lemma 11. The set $\mathcal{J}^*$ consists of all essential hyperedges.

▶ **Lemma 13.** *$\mathcal{J}^*$ contains all $J(\bar{e})$ where $\bar{e}$ is an essential hyperedge for (G, T, c).*

Proof. We fix an essential hyperedge $\bar{e}$ and show that $J(\bar{e})$ is in $\mathcal{J}^*$. To do this, we construct an independent set in $\mathcal{M}$ extended by $J(\bar{e})$, but not extended by $J(e)$ for any hyperedge $e \neq \bar{e}$. Let $\mathcal{T}$ be a terminal partition with $\mathsf{min\text{-}cut}_G(\mathcal{T}) \leq c$ such that every minimum multiway cut of $\mathcal{T}$ includes $\bar{e}$. We fix a minimum multiway cut F of $\mathcal{T}$ in G, and let $(V_0, \ldots, V_s)$ be the vertex partition according to $G \setminus F$. We assume that the component V_0 maximizes $|T \cap V_0|$ among $V_0, \ldots, V_s$, and V_1 maximizes $|E(V_1)|$ among $V_1, \ldots, V_s$. Additionally, the other components $V_2, \ldots, V_s$ are sorted in the decreasing order of $\mathsf{cap}_T(\cdot)$ values. Let E_{small} denote the union of $E(V_{i_0+1}), E(V_{i_0+2}), \ldots, E(V_s)$.

Then we consider the following sets whose union will be a witness showing that $J(\bar{e})$ is in any representative set of $\{J(e) \mid e \in E(G)\}$. Let A^{u} denote the subset $\{e^{\mathsf{u}} \mid e \in (E_{\mathsf{small}} \cup F) \setminus \{\bar{e}\}\}$ of the universe set of the uniform matroid on E of rank $\kappa + c$. Let A_i^{g} be the subset $\{e_i \mid e \in F \cup \partial(T \cap V_i)\}$ of the universe set of $\mathcal{M}_i$ for $i \in [0, i_0]$. We need to demonstrate three assertions: **(i)** A^{u} is extended by $\{(\bar{e})^{\mathsf{u}}\}$ in the uniform matroid on E of rank $\kappa + c$, **(ii)** A_i^{g} is extended by $\{\mathsf{sink}_i(\bar{e})\}$ in $\mathcal{M}_i$ for $i \in [0, i_0]$, and **(iii)** any $J(e)$ with $e \neq \bar{e}$ cannot extend $A = A^{\mathsf{u}} \cup (\cup_{i \in [0, i_0]} A_i^{\mathsf{g}})$ in $\mathcal{M}$. By combining these assertions, we can conclude that A is extended in $\mathcal{M}$ by $J(\bar{e})$ only, which completes the proof of Lemma 13.

Assertion (i). Recall that the size of F is at most c. Then A^{u} is extended by $\{(\bar{e})^{\mathsf{u}}\}$ in the uniform matroid on E of rank $\kappa + c$ if $|E_{\mathsf{small}}| \leq \kappa$ because $\bar{e}$ is not in $(E_{\mathsf{small}} \cup F) \setminus \{\bar{e}\}$.

We first show that $\sum_{j \in [s]} \mathsf{cap}_T(V_j) \leq (3d + 2rc)$. Note that every component V_j has at most d terminals of T for $j \in [s]$ because (G, T, c) is d-unbreakable. Precisely, if a component V_j has more than d terminals, then V_0 also contains more than d terminals since we chose V_0 as containing the most terminals, and thus, the d-unbreakable property is violated from $|V_j \cap T|$, $|T \setminus V_j| > d$. Let x be the smallest index in $[s]$ such that $\sum_{j \in [x]} |T \cap V_j| > d$. By choosing x in this manner, the total size of $T \cap V_j$ for all indices $1 \leq j < x$ is at most d, and for all indices $j > x$ is also at most d due to the d-unbreakable property. Due to $|T \cap V_x| \leq d$, we have $\sum_{j \in [s]} |T \cap V_j| \leq 3d$. Moreover, since each hyperedge in F can intersect at most r components and $\mathsf{cap}_T(V_j) = 2|\partial_G V_j| + |T \cap V_j|$, we have $\sum_{j \in [s]} \mathsf{cap}_T(V_j) \leq (3d + 2rc)$.

Now we focus on the components $V_{i_0+1}, \ldots, V_s$ as follows. Since (G, T, c) is α-dense, $|E(V_j)| \leq (\mathsf{cap}_T(V_j))^{\alpha}$ for $j \in [2, s]$ by the α-dense property.[1] Thus, the ordering $\mathsf{cap}_T(V_2) \geq \cdots \geq \mathsf{cap}_T(V_s)$ implies that $\mathsf{cap}_T(V_j) \leq (3d + 2rc)/(j - 1)$. Therefore, we have

$$|E_{\mathsf{small}}| = \sum_{j \in [i_0+1, s]} |E(V_j)| \leq \sum_{j \in [i_0+1, s]} \left(\frac{3d + 2rc}{i_0}\right)^{\alpha} \leq \sum_{j \in [i_0+1, s]} \left(\frac{d}{4}\right)^{\alpha}$$
$$\leq (d/4)^{\alpha} \cdot rc \leq (d/2)^{\alpha+1} \cdot (1/2)^{\alpha-1} r \leq (d/2)^{\alpha - i_0 - 2} = \kappa.$$

[1] $|E(V_j)| = \min\{|E(V_j)|, |E(V \setminus V_j)|\} \leq \mathsf{cap}_T(V_j)^{\alpha}$ for $j \in [2, s]$.

The inequalities follow from $i_0 = 30r$, $s \le rc$, and the assumption we made at the beginning of this subsection: $c \le d \le k$ and $\alpha \ge 35r \log d \ge (i_0 + 2) \log d$.[2] It implies $|A^{\mathsf{u}}| \le \kappa + c - 1$ due to $(\bar{e})^{\mathsf{u}} \notin A^{\mathsf{u}}$. Therefore, $\{(\bar{e})^{\mathsf{u}}\}$ extends A^{u} in the uniform matroid on E of rank $\kappa + c$.

Sketch of Assertions (ii-iii). Details are in the full version. Briefly, **Assertions (ii-iii)** holds since there are $|F| + 1$ pairwise edge-disjoint paths from $T \setminus V_i$ to F in G for each component V_i if we allow using $\bar{e} \in F$ twice while there is no path from $T \setminus V_i$ to $E(V_i)$ excluding F. Recall that $\bar{e}$ is essential. Therefore, $\{\mathsf{sink}_i(e)\}$ extends A_i^{g} if and only if $e = \bar{e}$ by Observation 12 along with the Menger's theorem for hypergraphs [26]. ◀

Lemma 14 holds by Lemma 11 and Lemma 13. The detailed proof is in the full version.

▶ **Lemma 14.** *If (G, T, c) is d-unbreakable and α-dense with $\alpha \ge 35r \log d$, $c \le d \le k$, and $m > kd^{\alpha-1}$, then there is a non-essential hyperedge.*

4.3 Non-Essential Hyperedge in a General Instance

In this section, we show that a minimal multicut-mimicking network of (G, T, c) has at most $|T|c^{O(r \log c)}$ hyperedges. For this achievement, we start by assuming that (G, T, c) has more than $|T|c^{\Omega(r \log c)}$ hyperedges, and find a subinstance that has a non-essential hyperedge. Note that the obtained hyperedge is also non-essential in (G, T, c) by Lemma 3, and thus, (G, T, c) is not minimal by Lemma 2. Note that we can find a $5c$-unbreakable subinstance (G', T', c) of $|T'|c^{\Omega(r \log c)}$ hyperedges [10]. Details are in the full version. In the following, we suppose that (G, T, c) is a $5c$-unbreakable with $|T|c^{\Omega(r \log c)}$ hyperedges.

We recursively find a subinstance until it satisfies the conditions in Lemma 14. Then it has a non-essential hyperedge, furthermore, it is also non-essential in the original instance. Note that constructing a subinstance might increase the size of a hyperedge by inserting two anchored terminals for a restricted hyperedge. However, it increases the hyperedge size only if the hyperedge has less than two terminals. Additionally, once increased hyperedge has two (anchored) terminals. Thus, the rank is increased by at most one even if we obtain a subinstance recursively. We fix r as the rank of the original instance to avoid confusion.

Construction of non-minimal instance. We suppose that (G, T, c) is d-unbreakable with $d = \min\{5c, |T|\}$. Then we show that if G has more than $|T|d^{\alpha(c)-1}$ hyperedges, (G, T, c) has a non-essential hyperedge by inductively along m, where $\alpha(c) = 35(r + 2) \log(5c)$. Recall that r is a fixed constant so that the rank of (G, T, c) is at most $r + 1$, and α is the constant derived from c only. For simplicity, we use α to denote $\alpha(c)$ when the context is clear. If (G, T, c) is α-dense, then it has a non-essential hyperedge by Lemma 14.

When (G, T, c) is not α-dense, there is a witness vertex set $X \subseteq V$ with $0 < |E(X)| \le |E(V \setminus X)|$, $|\partial X| \le c$, and $|E(X)| > (\mathsf{cap}_T(X))^{\alpha}$.[3] If $|X \cap T| > |T \setminus X|$, we *replace* X with $V \setminus X$. Note that the following inequalities still hold: $|\partial X| \le c$, $|E(X)| > (\mathsf{cap}_T(X))^{\alpha}$, and $|E(V \setminus X)| > 0$. The first one holds since $\partial X = \partial(V \setminus X)$, and the second one holds since $|T \cap X|$ and $\mathsf{cap}_T(X)$ are decreased by the replacement while $|E(X)|$ is increased. The last holds since we chose X so that $E(X), E(V \setminus X) \ne \emptyset$. Additionally, the size of $T \cap X$ is at most d since our instance is d-unbreakable. We move to the subinstance $(\hat{G}[X], T_X, c_X)$ with respect to X, where $c_X = \min\{c, |T_X|\}$ and T_X is the union of terminals $T \cap X$ and the anchored terminals. Recall that the size of T_X is at most $\mathsf{cap}_T(X; G) = |T \cap X| + 2|\partial X| \le d + 2c$. Lemma 15, proved in the full version, ensures the safeness of this inductive proof.

[2] Precisely, these assumption give us $r \cdot \left(\frac{1}{2}\right)^{\alpha-1} \le \left(\frac{d}{2}\right)^{-i_0-3}$ which implies the last inequality.

[3] (G, T, c) is α-dense if $|E(Y)| \le (\mathsf{cap}_T(Y))^{\alpha}$ for any $Y \subseteq V$ with $0 < |E(Y)| \le |E(V \setminus Y)|$ and $|\partial Y| \le c$.

▶ **Lemma 15.** *$(\hat{G}[X], T_X, c_X)$ is d_X-unbreakable with $d_X = \min\{5c_X, |T_X|\}$. Additionally, $\hat{G}[X]$ has more than $|T_X|d_X^{\alpha'-1}$ hyperedges but less than $|E(G)|$, where $\alpha' = \alpha(c_X)$.*

We recursively obtain a subinstance until it becomes α-dense. Note that k, d, m, and α change during the recursion while the rank is always at most $r+1$, where k and m denote the number of terminals and hyperedges, respectively, in the current instance. However, Lemma 15 guarantees that $m > kd^{\alpha-1}$ holds at each step. Moreover, m is strictly decreased. Thus, we always reach a d-unbreakable and α-dense instance satisfying the conditions of Lemma 14, and it has a non-essential hyperedge. It is easy to show that the hyperedge is also non-essential in the original instance by applying Lemma 3 recursively.

Therefore, a d-unbreakable instance (G, T, c) with $m > |T|d^{\alpha(c)-1}$ and $d = \min\{5c, |T|\}$ has a non-essential hyperedge. This section proves Theorem 10.

▶ **Theorem 10.** *Every minimal instance (G, T, c) has at most $|T|c^{O(r \log c)}$ hyperedges.*

References

1 Charles J. Alpert and Andrew B. Kahng. Recent directions in netlist partitioning: A survey. *Integration*, 19(1-2):1–81, 1995. doi:10.1016/0167-9260(95)00008-4.

2 Nikhil Bansal, Ola Svensson, and Luca Trevisan. New notions and constructions of sparsification for graphs and hypergraphs. In *Proceedings of the 60th Annual Symposium on Foundations of Computer Science (FOCS 2019)*, pages 910–928, 2019. doi:10.1109/FOCS.2019.00059.

3 Parinya Chalermsook, Syamantak Das, Yunbum Kook, Bundit Laekhanukit, Yang P. Liu, Richard Peng, Mark Sellke, and Daniel Vaz. Vertex sparsification for edge connectivity. In *Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*, pages 1206–1225, 2021. doi:10.1137/1.9781611976465.74.

4 Shiri Chechik and Christian Wulff-Nilsen. Near-optimal light spanners. *ACM Transactions on Algorithms (TALG)*, 14(3):1–15, 2018. doi:10.1145/3199607.

5 Chandra Chekuri and Chao Xu. Minimum cuts and sparsification in hypergraphs. *SIAM Journal on Computing*, 47(6):2118–2156, 2018. doi:10.1137/18M1163865.

6 Yu Chen and Zihan Tan. On $(1+\varepsilon)$-approximate flow sparsifiers. In *Proceedings of the 35th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2024)*, pages 1568–1605, 2024. doi:10.1137/1.9781611977912.63.

7 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 4(8). Springer, 2015. doi:10.1007/978-3-319-21275-3.

8 Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994. doi:10.1137/S0097539792225297.

9 Arnold Filtser and Shay Solomon. The greedy spanner is existentially optimal. In *Proceedings of the 35th ACM Symposium on Principles of Distributed Computing (PODC 2016)*, pages 9–17, 2016. doi:10.1145/2933057.2933114.

10 Han Jiang, Shang-En Huang, Thatchaphol Saranurak, and Tian Zhang. Vertex sparsifiers for hyperedge connectivity. In *Proceedings of the 30th Annual European Symposium on Algorithms (ESA 2022)*, pages 70:1–70:13, 2022. doi:10.4230/LIPICS.ESA.2022.70.

11 Wenyu Jin and Xiaorui Sun. Fully dynamic s-t edge connectivity in subpolynomial time. In *Proceedings of the 62nd Annual Symposium on Foundations of Computer Science (FOCS 2022)*, pages 861–872. IEEE, 2022.

12 Jörg Hendrik Kappes, Markus Speth, Gerhard Reinelt, and Christoph Schnörr. Higher-order segmentation via multicuts. *Computer Vision and Image Understanding*, 143:104–119, 2016. doi:10.1016/J.CVIU.2015.11.005.

13 Michael Kapralov, Robert Krauthgamer, Jakab Tardos, and Yuichi Yoshida. Spectral hypergraph sparsifiers of nearly linear size. In *Proceedings of the 62nd Annual Symposium on Foundations of Computer Science (FOCS 2022)*, pages 1159–1170, 2022.

14 Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. *Journal of the ACM*, 67(3):1–50, 2020. `doi:10.1145/3390887`.

15 Robert Krauthgamer and Ron Mosenzon. Exact flow sparsification requires unbounded size. In *Proceedings of the 34th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2023)*, pages 2354–2367, 2023. `doi:10.1137/1.9781611977554.CH91`.

16 Yang P Liu. Vertex sparsification for edge connectivity in polynomial time. In *Proceedings of the 14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*, pages 83:1–83:15, 2023. `doi:10.4230/LIPICS.ITCS.2023.83`.

17 Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. Deterministic truncation of linear matroids. *ACM Transactions on Algorithms (TALG)*, 14(2):1–20, 2018. `doi:10.1145/3170444`.

18 Yaowei Long and Thatchaphol Saranurak. Near-optimal deterministic vertex-failure connectivity oracles. In *Proceedings of the 63rd Annual Symposium on Foundations of Computer Science (FOCS 2022)*, pages 1002–1010, 2022. `doi:10.1109/FOCS54457.2022.00098`.

19 Anand Louis and Yury Makarychev. Approximation algorithms for hypergraph small set expansion and small set vertex expansion. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, page 339, 2014.

20 Dániel Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394–406, 2006. `doi:10.1016/J.TCS.2005.10.007`.

21 Dániel Marx. A parameterized view on matroid optimization problems. *Theoretical Computer Science*, 410(44):4471–4479, 2009. `doi:10.1016/J.TCS.2009.07.027`.

22 James Oxley. Matroid theory. In *Handbook of the Tutte Polynomial and Related Topics*, pages 44–85. Chapman and Hall/CRC, 2022.

23 Muhammet Mustafa Ozdal and Cevdet Aykanat. Hypergraph models and algorithms for data-pattern-based clustering. *Data Mining and Knowledge Discovery*, 9:29–57, 2004. `doi:10.1023/B:DAMI.0000026903.59233.2A`.

24 Magnus Wahlström. Quasipolynomial multicut-mimicking networks and kernels for multiway cut problems. *ACM Transactions on Algorithms (TALG)*, 18(2):1–19, 2022. `doi:10.1145/3501304`.

25 Zi-Ke Zhang and Chuang Liu. A hypergraph model of social tagging networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(10):P10005, 2010.

26 Alexander Aleksandrovich Zykov. *Hypergraphs*, volume 29(6). IOP Publishing, 1974.