



Cut Sparsification and Succinct Representation of Submodular Hypergraphs

Yotam Kenneth  

Weizmann Institute of Science, Rehovot, Israel

Robert Krauthgamer  

Weizmann Institute of Science, Rehovot, Israel

Abstract

In cut sparsification, all cuts of a hypergraph $H = (V, E, w)$ are approximated within $1 \pm \epsilon$ factor by a small hypergraph H' . This widely applied method was generalized recently to a setting where the cost of cutting each hyperedge e is provided by a splitting function $g_e : 2^e \rightarrow \mathbb{R}_+$. This generalization is called a submodular hypergraph when the functions $\{g_e\}_{e \in E}$ are submodular, and it arises in machine learning, combinatorial optimization, and algorithmic game theory.

Previous work studied the setting where H' is a reweighted sub-hypergraph of H , and measured the size of H' by the number of hyperedges in it. In this setting, we present two results: (i) all submodular hypergraphs admit sparsifiers of size polynomial in $n = |V|$ and ϵ^{-1} ; (ii) we propose a new parameter, called spread, and use it to obtain smaller sparsifiers in some cases.

We also show that for a natural family of splitting functions, relaxing the requirement that H' be a reweighted sub-hypergraph of H yields a substantially smaller encoding of the cuts of H (almost a factor n in the number of bits). This is in contrast to graphs, where the most succinct representation is attained by reweighted subgraphs. A new tool in our construction of succinct representation is the notion of deformation, where a splitting function g_e is decomposed into a sum of functions of small description, and we provide upper and lower bounds for deformation of common splitting functions.

2012 ACM Subject Classification Theory of computation \rightarrow Sparsification and spanners; Theory of computation \rightarrow Submodular optimization and polymatroids; Mathematics of computing \rightarrow Hypergraphs; Theory of computation \rightarrow Lower bounds and information complexity

Keywords and phrases Cut Sparsification, Submodular Hypergraphs, Succinct Representation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.97

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version:* <https://arxiv.org/abs/2307.09110>

Funding This research was partially supported by the Israel Science Foundation grant #1336/23, by a Weizmann-UK Making Connections Grant, by a Minerva Foundation grant, by the Israeli Council for Higher Education (CHE) via the Weizmann Data Science Research Center, and by a research grant from the Estate of Harry Schutzman.

1 Introduction

A powerful tool for many graph problems is sparsification, where an input graph is replaced by a small graph that preserves (perhaps approximately) certain properties, for example all the input graph's cuts [7] or its spectrum [42, 6, 26]. Downstream applications can then be executed on the small graph, which improves the overall running time, and the small graph can also be stored (or sent to another site) instead of the input graph, which improves the memory (or communication) requirements. The extensive research on cut sparsification has started with the seminal work of Benczúr and Karger on cuts in graphs [7], and was later extended to hypergraphs [29, 5, 10] and to directed hypergraphs [40, 9, 27, 36]. In recent



© Yotam Kenneth and Robert Krauthgamer;

licensed under Creative Commons License CC-BY 4.0

51st International Colloquium on Automata, Languages, and Programming (ICALP 2024).

Editors: Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson;

Article No. 97; pp. 97:1–97:17



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



months the study of sparsification has been extended to even more general objects such as semi-norms [25], matroid quotients [38], and linear codes [28]. We focus on sparsifying a generalized form of hypergraphs, as explained next.

In recent years, the notion of cuts in a weighted hypergraph $H = (V, E, w)$ has been generalized to a setting where each hyperedge $e \in E$ has a *splitting function* $g_e : 2^e \rightarrow \mathbb{R}_+$, such that $g_e(\emptyset) = 0$, and the *value* of a cut $S \subseteq V$ is defined as

$$\text{cut}_H(S) := \sum_{e \in E} g_e(S \cap e). \tag{1}$$

Associating every $e \in E$ with the *all-or-nothing* splitting function, given by $g_e^{\text{aon}} : S \mapsto w_e \cdot \mathbb{1}_{\{S \neq \emptyset, e\}}$, clearly models an ordinary hypergraph $H = (V, E, w)$, where the value of a cut is the total weight of hyperedges that intersect both sides; in fact, a simple extension can model a directed hypergraph. Such a generalized hypergraph $H = (V, E, g)$, where $g = \{g_e\}_{e \in E}$, is called a *submodular hypergraph* if all its splitting functions g_e are submodular. Recall that a set function $g : 2^e \rightarrow \mathbb{R}_+$ is *submodular* if

$$\forall S, T \subseteq e, \quad g(S \cup T) + g(S \cap T) \leq g(S) + g(T).$$

Submodular hypergraphs are useful in clustering data with higher-order relations that are not captured by ordinary hyperedges [31, 32, 44, 34, 45, 48]. For example, the *small-side splitting* function, given by $g_e^{\text{sm}} : S \mapsto \min(|S|, |e \setminus S|)$, is employed when unbalanced cuts are preferable. Cut functions of submodular hypergraphs were studied also under a different name of *decomposable submodular functions*. A submodular function $f : 2^V \rightarrow \mathbb{R}_+$ is called *decomposable* if it can be written as $f = \sum_i f_i$, where each $f_i : 2^V \rightarrow \mathbb{R}_+$ is submodular. This notion is widely applied in data summarization [23, 33, 43], where each f_i is a submodular similarity function, and the task of summarizing the data under a given budget k is modeled by maximizing $f(S)$ over all $S \subset V$ of size $|S| \leq k$. Decomposable submodular functions arise also in welfare maximization, where each agent has a submodular utility function, for instance in approximation algorithms [16, 17] and in truthful mechanisms [15, 4].

We study how to succinctly represent all the cuts of a submodular hypergraph H up to $1 \pm \epsilon$ factor. We examine two complementary approaches: (1) *sparsification*, which reduces the number of hyperedges, i.e., H is represented using a sparse H' ; and (2) *deformation*, which replaces large hyperedges or complicated splitting functions by new ones of low space complexity, i.e., H is represented using H' whose hyperedges can be stored succinctly. These approaches can yield (separately and/or together) a sparsifier H' that can be encoded using a small number of bits. More generally, we may consider a general encoding that *need not* rely on a sparsifier H' , e.g., an explicit list of all the $2^{|V|}$ cut values.

Let us introduce some basic notation to make the discussion more precise. Throughout, let $n := |V|$; we write $\tilde{O}(t)$ or $\tilde{\Omega}(t)$ to suppress a polylogarithmic factor in t , and $O_\alpha(t)$ or $\Omega_\alpha(t)$ to hide a factor that depends only on α .

► **Definition 1.1** (Sparsifier). *A cut sparsifier of quality $1 + \epsilon$ for $H = (V, E, g)$, or in short a $(1 + \epsilon)$ -sparsifier, is a submodular hypergraph $H' = (V, E', g')$ such that*

$$\forall S \subseteq V, \quad \text{cut}_{H'}(S) \in (1 \pm \epsilon) \cdot \text{cut}_H(S). \tag{2}$$

The size of the sparsifier is $|E'|$. We call H' a reweighted subgraph of H if $E' \subseteq E$ and each function g'_e for $e \in E'$ is a scaling of g_e (i.e., $g'_e \equiv s_e g_e$ for some $s_e > 0$).

► **Question 1.2** (Sparsification). *Do all submodular hypergraphs admit a reweighted-subgraph sparsifier with few hyperedges, say $\text{poly}(\epsilon^{-1}n)$? And which families of splitting functions admit even smaller sparsifiers, like $\tilde{O}_\epsilon(n^2)$ or even $\tilde{O}_\epsilon(n)$?*

The first question (about a polynomial bound) was previously answered for several families of splitting functions (see Section 1.1 for a detailed account), but despite this significant progress, the case of general submodular splitting was left open in [39], where the bound on the sparsifier size depends on g and is exponential in n in the worst case. We answer this first question in the affirmative, and also address the second question by showing families of splitting functions that admit even smaller sparsifiers.

We further ask about a more general notion, of encoding an approximation of all the cuts of H , which can potentially be more succinct than a sparsifier.

► **Question 1.3 (Succinct Representation).** *What is the smallest encoding (in bits of space) that stores a submodular hypergraph H so as to report $(1 + \epsilon)$ -approximation to every cut value? In particular, what is the smallest number of bits $s = s(\epsilon, n)$ that suffices to store a sparsifier for H ?*

For simplicity, we ask above only about the existence of a sparsifier or an encoding, but we are of course interested also in fast algorithms to build them. Fortunately, an algorithmic solution follows from the existential ones because our proofs are constructive. Furthermore, the running times are polynomial under the assumption that every g_e takes integral values and $\max_{S \subseteq e} g_e(S) \leq \text{poly}(n)$.¹

1.1 Sparsification: All Submodular Hypergraphs

We start with addressing Question 1.2. Our first result (proved in Section 2) provides the first polynomial (in n) bound for all submodular splitting functions; the previous bound, due to [39], was $O_\epsilon(n^2 B_H)$, where $B_H := \max_{e \in E} |\mathcal{B}(g_e)|$ and $\mathcal{B}(g_e)$ is the set of extreme points in the polytope of g_e .² In general, B_H can be exponential in n , for example small-side splitting g_e^{sml} has $|\mathcal{B}(g_e^{\text{sml}})| = 2^{\Theta(|e|)}$.

► **Theorem 1.4.** *Every submodular hypergraph admits a $(1 + \epsilon)$ -sparsifier of size $O(\epsilon^{-2} n^3)$, which is in fact a reweighted sub-hypergraph.*

This bound is within factor $O_\epsilon(n)$ of the $\Omega(n^2/\epsilon)$ lower bound known for cut sparsification of directed hypergraphs [36]. We also show that if all the splitting functions are monotone (i.e., $g_e(S) \leq g_e(T)$ for all $S \subseteq T$), then the sparsifier size can be improved to $O_\epsilon(n^2)$. Monotone submodular functions arise in many applications, however no sparsification bound was previously known for this family.³ The formal statement and its proof appear in Section 2.

Related Work. Previous work on sparsification focused mostly on specific splitting functions. The study of this problem began with sparsifiers for undirected graph cut; the current size bound is $O(\epsilon^{-2} n)$ edges [6], which improves over [7] and is known to be tight [2, 8]. Furthermore, sparsifiers of size $\tilde{O}_\epsilon(n)$ are known for all-or-nothing splitting g_e^{aon} [10] (see also [38]) and for *product splitting*, given by $g_e^{\text{prd}} : S \mapsto |S| \cdot |e \setminus S|$ [13]. In contrast, for the splitting that models cuts in a directed hypergraph, the best construction known has size $\tilde{O}_\epsilon(n^2)$ [36], which is near-tight with an $\Omega(n^2/\epsilon)$ lower bound [36]; this function, called

¹ The running times of Theorem 1.4 and Theorem 1.9 are polynomial in general. Theorem 1.6 is polynomial under the stated assumption.

² A recent manuscript [30] claims that the proof in [39] has a flaw and holds only for monotone submodular hypergraphs.

³ The running time of [39] was improved in [30], where a sparsifier of size $O(\epsilon^{-2} n^2 B)$ for monotone functions with low curvature is constructed in polynomial time.

directed all-or-nothing splitting, is given by $g_e^{\text{d-aon}} : S \mapsto \mathbb{1}_{\{e_T \cap S \neq \emptyset \wedge e_H \not\subseteq S\}}$, where $e_H, e_T \subseteq e$ are the hyperedge's head and tail, respectively. A recent result is more general and shows that the entire family of symmetric splitting functions admits sparsifiers of size $\tilde{O}_\epsilon(n)$ [25].

Figure 1 depicts several families of splitting functions and the sparsification bounds known for them, including our results from above and from Section 1.2.

Techniques. Our sparsification method follows the importance-sampling approach, which has been used extensively in the literature. Every hyperedge $e \in E$ is assigned an importance σ_e , and sampled with probability p_e that is (at least) proportional to σ_e , and the splitting function of every sampled e is scaled by $1/p_e$. The expected sparsifier size is clearly proportional to $\sum_{e \in E} \sigma_e$.

A standard method to set the importance of a hyperedge $e \in E$, is to consider all its possible cuts, namely, $\sigma_e := \max_{S \subseteq V} g_e(S \cap e) / \text{cut}_H(S)$, and this method was indeed used in [39]. Bounding $\sum_{e \in E} \sigma_e$ naively by replacing the maximization over $S \subseteq V$ by summation yields an exponential size bound. An improved bound was given in [39] based on a quantity B_H related to the polytopes of the splitting functions. Unfortunately, this improved bound is still exponential for many families of splitting functions.

Our main contribution is to identify a set of “basic” quantities for each hyperedge e that can serve as coarse approximations of its splitting function g_e . These approximations allow us to define new sampling probabilities and achieve an improved size bound: Given $e \in E$, define the minimum directed cut between $u, v \in V$ to be $g_e^{u \rightarrow v} := \min_{S \subseteq V: u \in S, v \notin S} g_e(S \cap e)$;⁴ then our main technical lemma bounds $g_e(\cdot)$ from below and from above by

$$\forall S \subseteq V, \quad \max_{u \in S, v \in V \setminus S} g_e^{u \rightarrow v} \leq g_e(S \cap e) \leq \sum_{u \in S, v \in V \setminus S} g_e^{u \rightarrow v}. \quad (3)$$

The lower bound holds by definition, and the upper bound is analogous to bounding the value of a graph cut by the sum of the maximum flows between all pairs of vertices across the cut. It is well-known that importance sampling will produce a sparsifier even if σ_e is replaced with an over-estimate for it. We replace σ_e with $\rho_e := \sum_{(u,v) \in V \times V} g_e^{u \rightarrow v} / \sum_{f \in E} g_f^{u \rightarrow v}$, which we can easily see is an over-estimate, i.e., $\rho_e \geq \sigma_e$, by using the two bounds from (3) to verify that

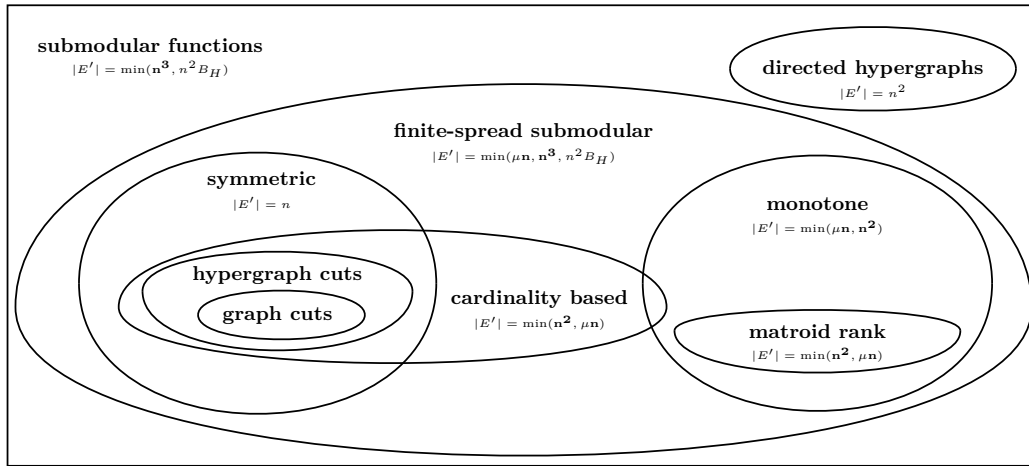
$$\forall S \subseteq V, \quad \frac{g_e(S \cap e)}{\text{cut}_H(S)} = \frac{g_e(S \cap e)}{\sum_{f \in E} g_f(S \cap f)} \leq \sum_{u \in S, v \in V \setminus S} \frac{g_e^{u \rightarrow v}}{\sum_{f \in E} g_f^{u \rightarrow v}} \leq \rho_e.$$

The expected number of hyperedges in the sparsifier H' equals to $\sum_{e \in E} \rho_e$ times an amplification factor M , where $M = O(\epsilon^{-2}n)$ is sufficient by standard arguments (combining a concentration bound and a union bound). The crux here is that it is easy to bound $\sum_{e \in E} \rho_e \leq O(n^2)$, basically swapping the order of a double summation. Another advantage of ρ_e is that it can be computed in polynomial time, while computing σ_e requires maximizing the ratio of two submodular functions, which is NP-hard in general.

In the monotone case, we follow the same approach but employ a simpler over-estimate $\rho'_e := \sum_{v \in e} g_e(\{v\}) / \text{cut}_H(\{v\})$. The proof is similar to the general case, except that instead of (3) we use the straightforward bound

$$\forall S \subseteq V, \quad \max_{v \in S} g_e(\{v\} \cap e) \leq g_e(S \cap e) \leq \sum_{v \in S} g_e(\{v\} \cap e).$$

⁴ The most natural case is $u, v \in e$, but considering all $u, v \in V$ streamlines the presentation.



■ **Figure 1** Sparsification bounds for various families of submodular functions, omitting for simplicity $\text{poly}(\epsilon^{-1} \log n)$ factors.

1.2 Sparsification: Parameterized by Spread

We already know that submodular splitting functions can have very different optimal sparsification bounds, see e.g. the bounds $\tilde{\Theta}_\epsilon(n)$ and $\tilde{\Theta}_\epsilon(n^2)$ mentioned above. However, there are too many submodular functions to analyze each one separately, and we thus seek a parameter that can control the sparsifier size. Our approach is inspired by the notion of imbalance in a directed graph $G = (V, E, w)$, defined as the worst ratio between antiparallel edge weights, i.e., $\beta_G := \max\{w(i, j)/w(j, i) : i, j \in V\}$. This parameter can be used to show that every directed graph admits a sparsifier of size $\tilde{O}_\epsilon(\beta_G n)$.⁵ For submodular hypergraphs, we propose an analogous parameter, which is basically the ratio between the maximum and minimum values of the splitting function, excluding certain trivial cuts.

► **Definition 1.5 (Spread).** For hyperedge $e \in E$ with splitting function g_e , let $W_e := \{\emptyset\}$, unless $g_e(e) = 0$ in which case $W_e := \{\emptyset, e\}$. The spread of e is

$$\mu_e := \frac{\max_{T \subseteq e} g_e(T)}{\min_{S \subseteq e: S \notin W_e} g_e(S)}. \tag{4}$$

Our third result (proved in the full version) constructs a sparsifier whose size depends on the spread of the input H , defined as $\mu_H := \max_{e \in E} \mu_e$. By convention, the spread μ_e is called *finite* if it is well-defined (the denominator in (4) is non-zero), and similarly μ_H is called finite if it is well-defined (all the terms μ_e are finite).

► **Theorem 1.6 (Sparsifier Parameterized by Spread).** Every submodular hypergraph $H = (V, E, g)$ with finite spread admits a $(1 + \epsilon)$ -sparsifier of size $\tilde{O}(\epsilon^{-2} \mu_H n)$, which is a subweighted-subgraph.

Many natural submodular functions have finite spread, and in many common cases even $\mu_H \leq n$. This can be seen, for example, in an easy application of Theorem 1.6 to approximation of coverage functions, see the full version for details. Another example is the

⁵ This condition can actually be relaxed significantly to $\beta_G := \max\{\text{cut}_G(S)/\text{cut}_G(\bar{S}) : S \subset V\}$, and the same sparsification bound still holds [9].

sparsification of the capped version of small-side splitting, given by $g_e : S \mapsto \min(|S|, |e \setminus S|, K)$ for $K > 0$, which clearly has spread $\mu_e \leq K$. This function is part of a much larger family, cardinality-based splitting functions, a notion formalized in [46] as follows: A submodular function $g_e : 2^e \rightarrow \mathbb{R}_+$ is called *cardinality-based* if there exists a function $f_e : [|e|] \rightarrow \mathbb{R}_+$ such that $g_e : S \mapsto f_e(|S|)$. Cardinality-based functions, which are commonly used in submodular hypergraph clustering, all have spread $\mu_e \leq n$, which is an easy consequence of the symmetry and subadditivity of g_e . By Theorem 1.6, these splitting function admit a $(1 + \epsilon)$ -sparsifier of size $\tilde{O}(\epsilon^{-2}n^2)$, which is the first bound for this family.

It is easily verified that for monotone splitting functions, the spread is approximately equal to the imbalance, when we generalize the imbalance from above to hyperedges by $\beta_e := \max\{g_e(S)/g_e(e \setminus S) : S \subset V\}$.⁶ Hence, we immediately obtain the following.

► **Corollary 1.7.** *Every finite-spread monotone splitting function admits a $(1 + \epsilon)$ -sparsifier of size $\tilde{O}(\epsilon^{-2}\beta_H n)$.*

Two other examples of commonly used monotone functions with finite spread are set-coverage functions (defined in the full version) and the *matroid-rank functions*,⁷ which have $\mu_e = r$ where r is the rank of the matroid.

We remark that spread does not fully characterize the sparsifier size. Indeed, symmetric functions can have a large spread μ_e but still admit $\tilde{O}_\epsilon(n)$ sparsifier due to [25], consider e.g. product splitting g_e^{prd} which has $\mu_e = O(n)$. Furthermore, directed all-or-nothing splitting $g_e^{\text{d-aon}}$ does not have finite spread, and nevertheless admits a sparsifier of size $\tilde{O}_\epsilon(n^2)$ [36]. Figure 1 depicts different families of splitting functions including that of finite spread, and the sparsification bounds known for them.

Techniques. Our technique is based on approximate H as an undirected hypergraph and use the sampling probabilities of [10] but amplified by μ_e for each hyperedge. This is a known technique in generalizing sampling mechanisms. Our main contribution is to identify the spread as a relevant and useful parameter. We remark that the generalization of balance, which is known to control the size of sparsifier in directed graphs, to submodular hypergraphs does not suffice for sparsification. Furthermore, we prove that the spread also characterizes other traits of splitting function, such as the deformation lower bound.

1.3 Succinct Representation

We provide the first example of submodular splitting functions for which sparsifiers that are not subgraphs are provably (much) more succinct than sparsifiers that are reweighted subgraphs.⁸ To be more precise, we exhibit a natural family of splitting functions, where the former $(1 + \epsilon)$ -sparsifiers take only $\tilde{O}_\epsilon(n)$ bits (Corollary 1.10), while the latter $(1 + \epsilon)$ -sparsifiers require $\tilde{\Omega}_\epsilon(n^2)$ bits (Theorem 1.11). It follows that a reweighted subgraph *need not be* the smallest encoding that stores a $(1 + \epsilon)$ -approximation of the cuts values, and by a wide margin!

⁶ For a monotone g_e , the spread is $\mu_e = g_e(V) / \min_{v \in V} g_e(\{v\})$ and the imbalance is $\beta_e = \max_{v \in V} g_e(V \setminus \{v\}) / g_e(\{v\})$, and they differ by at most a constant factor by the subadditivity of g_e .

⁷ For a matroid with ground set e and independent sets \mathcal{G} , the rank function is given by $g_e : S \mapsto \max_{T \subseteq S: T \in \mathcal{G}} |T|$. This rank function is submodular and monotone.

⁸ Previously, a non-subgraph sparsifier was shown in [1] for small-side splitting, however it optimizes the number of hyperedges and not the encoding size.

Our plan for constructing a succinct representation has two stages. The first stage creates a $(1 + \epsilon)$ -sparsifier H' , by deforming each $e \in E$ into multiple small hyperedges. The second stage computes for this H' a $(1 + \epsilon)$ -sparsifier H'' that is a reweighted subgraph. It then follows that H'' is a $(1 + \epsilon)^2$ -sparsifier, and has a few hyperedges that are all small.

► **Definition 1.8.** A splitting function $g_e : 2^e \rightarrow \mathbb{R}_+$ on hyperedge e is called $(1 + \epsilon)$ -approximable with support size p if there are submodular functions $g_{e_i} : 2^{e_i} \rightarrow \mathbb{R}_+$ for $i = 1, \dots, r$, each on a hyperedge $e_i \subseteq e$ of size $|e_i| \leq p$, such that

$$\forall S \subseteq e, \quad \sum_{i=1}^r g_{e_i}(S \cap e_i) \in (1 \pm \epsilon)g_e(S).$$

Our example is the family of *additive* splitting functions, defined as functions g_e that can be written as either $g_e : S \mapsto \min(|S|, K)$ or $g_e : S \mapsto \min(|S|, |e \setminus S|, K)$ for some $K > 0$. The next theorem (proved in the full version) achieves the first stage in our plan above; it shows that additive functions can be $(1 + \epsilon)$ -approximated by creating several copies of e and sampling the vertices.

► **Theorem 1.9** (Deformation of Additive Functions). *Let g_e be an additive splitting function on hyperedge e . Then g_e can be $(1 + \epsilon)$ -approximated with support size $O(\epsilon^{-2}(|e|/K) \log |e|)$.*

Following our plan, suppose that given an input H , we first apply Theorem 1.9 to obtain a sparsifier H' with small support size. The construction of H' also implies that it has small spread, $\mu_{H'} \leq O(\epsilon^{-2} \log n)$. Applying Theorem 1.6 on H' we obtain a succinct representation H'' . A straightforward encoding of H'' then proves the following corollary (see the full version for details).

► **Corollary 1.10** (Additive Functions admit Small Representation). *Let $H = (V, E, \{g_e\})$ be a submodular hypergraph such that every g_e is additive with parameter $K_e > 0$, and let $\hat{K} := \min_{e \in E} K_e/|e|$ be a normalized bound on K_e over all hyperedges. Then H admits a $(1 + \epsilon)$ -sparsifier with encoding size $O(\epsilon^{-6} \hat{K}^{-1} n \log^4 n)$ bits.*

The next theorem (proved in the full version) shows that reweighted-subgraph sparsifiers of additive functions require $\Omega(n^2)$ bits in the worst-case. Putting this together with our succinct representation from Corollary 1.10, we conclude that relaxing the (natural) restriction to reweighted subgraphs improves the space complexity by a factor of $\tilde{\Omega}_\epsilon(n\hat{K})$, observe that this can be $\tilde{\Omega}_\epsilon(n)$ when $\hat{K} \in \Omega(1)$.

► **Theorem 1.11** (Reweighted Sparsifiers Require $\Omega(n^2)$ Bits). *There exists a family \mathcal{H} of hypergraphs with additive splitting functions with parameter $1 \leq K \leq n/3$, such that encoding a reweighted-subgraph $(1 + \epsilon)$ -sparsifier for an input $H \in \mathcal{H}$ requires $\Omega(n^2)$ bits.*

This lower bound is surprising because in the case of undirected graphs, the best encoding size is achieved by a reweighted-subgraph sparsifier [6, 2, 8]. Our proof is based on a technical lemma that can be applied to many cardinality-based splitting functions. Furthermore, Theorem 1.11 can be extended to the directed all-or-nothing splitting function $g_e^{\text{d-aon}}$, to show a lower bound of $\Omega(n^3/\epsilon)$ bits. For details see the full version.

Finally, we can also prove a space lower bound for an arbitrary encoding of cuts in a directed hypergraph (arbitrary means that it need not represent a reweighted-subgraph sparsifier, see the full version for details). This proof provides an ϵ^{-1} factor improvement over the trivial lower bound of $\Omega(n^2)$ bits. The proof combines the techniques from Theorem 1.11 with a lower bound from [36] on the number of edges in a reweighted-subgraph sparsifier.

► **Theorem 1.12.** *There exists a family of directed hypergraphs \mathcal{H} such that encoding a $(1 + \epsilon)$ -approximation of their cuts requires $\Omega(n^2/\epsilon)$ bits.*

Techniques. Our lower bound for the encoding size of reweighted-subgraph sparsifiers (Theorem 1.11) boils down to a counting argument on a large family of hypergraphs \mathcal{H} , that have sufficiently different cut values and thus require distinct encodings. We construct hypergraphs in this family \mathcal{H} by partitioning the vertices into three parts V, U, W , and adding hyperedges that contain vertices from all three parts. We first create hyperedges consisting of a large random subset of vertices from V ; this adds entropy that will differentiate between hypergraphs in \mathcal{H} . We then augment each hyperedge with vertices from U , where each hyperedge is defined by a word in the Hadamard code. We use the structure of this code to show that by making cut queries to a hypergraph $H \in \mathcal{H}$, one can recover the random bits encoded in the adjacency matrix of H induced on V . We use W to create an unsparsifiable hypergraph, i.e., one where removing any hyperedge will violate the approximation guarantee. Finally, every hyperedge on $V \cup U$ is combined with a hyperedge on W .

1.4 Deformation Lower Bounds

Our success in finding a small succinct representation for additive functions motivates searching for deformations of other splitting functions.

A similar problem, of approximating a submodular function by functions of small support but over the uniform distribution (i.e., in average-case rather than worst-case), has received significant attention [19, 11, 24, 18, 20], and it is known that every submodular function $f : 2^V \rightarrow [0, 1]$ can be approximated within additive error ϵ using support size $O(\epsilon^{-2} \log \epsilon^{-1})$ [20]. We show (see the full version) that a similar result is unfortunately not possible in our setting (multiplicative error for worst-case approximation).

► **Theorem 1.13** (Approximation Requires Large Support Size). *Let g_e be an additive splitting function on a hyperedge e . Then every 1.1-approximation of g_e must have support size $p \geq \Omega(|e|/K)$.*

Techniques. The proof of Theorem 1.13 is based on a technical lemma that can be applied to many splitting functions. The main idea is to examine a certain quantity δ_t , which is related to the notion of curvature (of a submodular function). The curvature is often used to parameterize approximation guarantees in maximization of submodular optimization [12, 47]. Intuitively, both the curvature and δ_t characterize the locality of the function, i.e., how much error is introduced by decomposing the function into smaller parts and summing them. The main difference between the two quantities is that the curvature looks at the marginal contributions and δ_t characterizes the curvature of the union of two sets of size t . Furthermore, in the approximation setting, a low worst-case curvature is desirable while for our proof it suffices that δ_t is high for many sets of size t . Specifically, we show that if a constant fraction of pairs of subsets of size t have constant positive δ_t , then g_e cannot be approximated with support size smaller than $O(\delta_t^2 n/t)$.

By applying the technical lemma, we obtain lower bounds on the support size required to approximate several natural splitting functions, as presented in Table 1.

1.5 Related Work

Submodular functions appear in many applications, and have been studied extensively in the literature. In particular, the problem of finding a simple representation for submodular functions has been studied in several works. An $O(\sqrt{n} \log n)$ -approximation for monotone submodular functions by functions of the form $f(S) = \sqrt{\sum_{v \in S} c_v}$, where $c_v > 0$ are weights for all $v \in V$, was obtained in [22]. A later result [14] showed the same approximation using

■ **Table 1** Our lower bounds on the support size for several families of splitting functions. They are all obtained by applying the technical lemma, stated for simplicity for sufficiently small fixed $\epsilon > 0$ and $|e| = n$.

Function Family	Example	Support Size	See
additive functions	$g_\epsilon(S) = \min(S , K)$	$\Omega(n/K)$	Lemma 1.13
polynomial	$g_\epsilon(S) = S ^\alpha$ for $\alpha \in (0, 1)$	$\Omega(n)$	The full version
logarithmic	$g_\epsilon(S) = \log(S + 1)$	$\Omega(n)$	The full version
cardinality based	$g_\epsilon(S) = f(S)$ for concave f	$\Omega(n/\mu_\epsilon^{1.5})$	The full version
unweighted	$g_\epsilon(v) = 1$ for all $v \in V$	$\Omega(n/\mu_\epsilon^3)$	The full version

coverage and budget-additive functions. The same paper also provided a lower bound of $\Omega(n^{1/3} \log^{-2} n)$ for approximating monotone submodular functions by coverage and budget additive. Approximating the all-or-nothing splitting function on n vertices using hyperedges with the all-or-nothing function and with support size r must incur approximation factor $\Omega(n/r)$ [37, Section 2.3].

It was previously shown that every symmetric cardinality-based splitting functions can be deformed into a sum of $|e|/2$ hyperedges with capped small-side splitting function, while preserving the value of g_ϵ exactly [46]. Subsequent work by the same authors [45], achieves a similar deformation but with $(1 + \epsilon)$ -approximation and using only $O(\epsilon^{-1} \log |e|)$ hyperedges. Notice the difference from our work, which focuses on an approximation with small support size.

1.6 Concluding Remarks

Our work provides several promising directions for future work. We prove that all submodular hypergraph admit sparsifiers of polynomial size (Theorem 1.4), leaving a gap of $\tilde{\Omega}_\epsilon(n)$ between the upper and lower bounds. We conjecture that submodular hypergraphs admit the same sparsification bounds as (the special case of) directed hypergraphs.

► **Conjecture 1.14.** *Every submodular hypergraph admits a $(1 + \epsilon)$ -sparsifier of size $O(\epsilon^{-2} n^2)$, which is in fact a reweighted sub-hypergraph.*

Notice that the known lower bound of $\Omega(n^2/\epsilon)$ is not tight with this conjecture, and improving it is an interesting open problem. The main challenge in bridging the gap between our upper bound in Theorem 1.4 and the conjecture is the use of a union bound over all 2^n cuts. This challenge was overcome in graph and hypergraph sparsification by different methods, such as cut counting [7, 21, 10, 28], a matrix Chernoff bound [41], and chaining which uses progressively finer discretizations [5, 27, 36, 25]. Unfortunately, the matrix Chernoff bound is based on linear-algebra tools that are clearly inapplicable to hypergraphs. The cut-counting methods partition the cuts so that a union bound can be applied separately on each part; however these partitions rely on the binary nature of the all-or-nothing splitting function, which seems challenging in the submodular hypergraph setting, because the same g_ϵ can contribute very different values to different cuts $S \subseteq V$. The chaining methods seem more promising, especially the recent one [25] for all symmetric submodular functions, in which the contribution of a single g_ϵ is not binary, although it seems to rely on the splitting functions being symmetric.

In the sparsification setting, we obtain smaller sparsifiers for several families (monotone and finite-spread), however characterizing the optimal sparsifier size for each family remains open. In the succinct-representation setting, we found a useful deformation only for additive splitting functions (Theorem 1.9), and it would be desirable to find deformations for more families.

Another interesting avenue is to find applications or connections to other problems. For example, we show that Theorem 1.6 can be used to approximate a set-coverage function using a small ground set. Another potential application is constructing succinct representations for terminal cuts in a graph, see the full version for details on both applications.

2 Polynomial-Size Sparsifiers for Submodular Hypergraphs

This section proves Theorem 1.4 and its improvement in the monotone case. Our sparsification method is based on importance sampling, where hyperedges are sampled with probability that is (at least) proportional to their maximum relative contribution to any cut. A standard choice, that was indeed used in [39], is to sample every $e \in E$ with probability exactly proportional to its importance, defined as

$$\sigma_e := \max_{S \subseteq V} \frac{g_e(S \cap e)}{\sum_{f \in E} g_f(S \cap f)}.$$

The expected size of this sparsifier is proportional to the total importance $\sum_{e \in E} \sigma_e$, which is non-trivial to bound (e.g., naively replacing the maximization over $S \subseteq V$ by summation yields an exponential size bound). An improved bound on the size of a sparsifier constructed in this manner is given in [39], based on a quantity B_H related to the polytopes of the splitting functions. Unfortunately, this improved bound is still exponential for many families of splitting functions.

Our approach achieves a polynomial bound by using a different set of sampling probabilities and a different analysis. Our main insight is that it suffices to consider only a few cuts. Formally, define the minimum directed cut of g_e between $(u, v) \in V \times V$ as

$$g_e^{u \rightarrow v} := \min_{S \subseteq V: u \in S, v \notin S} g_e(S \cap e). \quad (5)$$

Notice that we do not require $u, v \in e$; clearly, $g_e^{u \rightarrow v} = 0$ if $u \notin e$, but $g_e^{u \rightarrow v}$ can be positive if $v \notin e$. Our sampling probabilities are proportional to

$$\rho_e := \sum_{(u,v) \in V \times V} \frac{g_e^{u \rightarrow v}}{\sum_{f \in E} g_f^{u \rightarrow v}},$$

where by convention the fraction is equal to zero if the denominator (and thus also the numerator) is zero. The proof follows by showing that $\rho_e \geq \sigma_e$, hence sampling every $e \in E$ with probability proportional to ρ_e suffices to approximate the cuts, and that the expected number of hyperedges in the sparsifier $O(\epsilon^{-2}n^3)$. Since $\rho_e \geq \sigma_e$, our analysis implies that the same size bound holds also for sampling with probabilities proportional to σ_e , i.e., for the sparsifier of [39] but with our amplification factor $M = O(\epsilon^{-2}n)$.

Finally, observe that the directed minimum cuts $g_e^{u \rightarrow v}$ can be computed in polynomial time using standard submodular minimization techniques [35].⁹ In contrast, calculating σ_e requires maximizing the ratio of two submodular functions, which is NP-hard. In the monotone case, previous work had achieved a polynomial running time [39, 30].

Proof of Theorem 1.4. Our construction of a quality $(1 + \epsilon)$ -sparsifier for H utilizes the importance sampling method, where each hyperedge is sampled independently with probability p_e that is defined below, and the splitting functions of every sampled hyperedge d is scaled by factor $1/p_e$.

⁹ In fact, computing an $O(1)$ -approximation to ρ_e would suffice, and this may be used to speed up the computation, at the cost of increasing the sparsifier size only by a constant factor.

We will use the following claim to bound cuts of H by minimum directed cuts. Throughout, we denote $\bar{S} = V \setminus S$.

▷ **Claim 2.1.** For every $e \in E$ and $S \subseteq V$,

$$\max_{u \in S, v \in \bar{S}} g_e^{u \rightarrow v} \leq g_e(S \cap e) \leq \sum_{u \in S} \sum_{v \in \bar{S}} g_e^{u \rightarrow v}.$$

The proof of Claim 2.1 appears later. Intuitively, it is similar to bounding the capacity of a cut in a graph by the sum of maximum flows between each vertex from S and each vertex from \bar{S} . We proceed assuming this claim, to show that $\rho_e \geq \sigma_e$.

► **Corollary 2.2.** For every $e \in E$ and $S \subseteq V$, we have $\rho_e \geq g_e(S \cap e) / \text{cut}_H(S)$.

Proof. By Claim 2.1, using both the upper bound and the lower bound on $g_e(\cdot)$,

$$\frac{g_e(S \cap e)}{\text{cut}_H(S)} = \frac{g_e(S \cap e)}{\sum_{f \in E} g_f(S \cap f)} \leq \sum_{u \in S, v \in \bar{S}} \frac{g_e^{u \rightarrow v}}{\sum_{f \in E} g_f^{u \rightarrow v}} \leq \rho_e.$$

Note that the first inequality holds even if $\text{cut}_H(S) = 0$, by our convention that if the denominator (and thus also numerator) is zero then the fraction is zero. ◀

For every hyperedge $e \in E$, set $\rho'_e := g_e(e) / \sum_{f \in E} g_f(f)$ as the importance of the cuts that contain the entire hyperedge (the case $S = V$), and let $p_e := \min(1, M(\rho_e + \rho'_e))$ for a suitable parameter $M = O(\epsilon^{-2}n)$. Now sample every hyperedge $e \in E$ independently with probability p_e and rescale the splitting functions of every sampled hyperedge by factor $1/p_e$. Let H' be the resulting hypergraph.

We first prove that the number of hyperedges in the sparsifier H' is $O(Mn^2)$, which satisfies the claimed size bound by our choice of $M = O(\epsilon^{-2}n)$. Let I_e be an indicator for the event that the hyperedge e is sampled into H' . The expected number of sampled hyperedges is

$$\begin{aligned} \mathbb{E} \left[\sum_{e \in E} I_e \right] &= \sum_{e \in E} p_e \leq M \sum_{e \in E} \left(\frac{g_e(e)}{\sum_{f \in E} g_f(f)} + \sum_{(u,v) \in V \times V} \frac{g_e^{u \rightarrow v}}{\sum_{f \in E} g_f^{u \rightarrow v}} \right) \\ &\leq M \left(1 + \sum_{(u,v) \in V \times V} \frac{\sum_{e \in E} g_e^{u \rightarrow v}}{\sum_{f \in E} g_f^{u \rightarrow v}} \right) \leq Mn^2, \end{aligned}$$

where the second inequality follows by changing the order of summation and the last one is because $|V \times V| = n^2$, but we can exclude from the summation the case $u = v$ (as it contributes 0 by our convention). By Markov's inequality, with high constant probability the sparsifier has at most $O(Mn^2)$ hyperedges.

Let us prove that the sparsifier H' indeed approximates the cuts of H . Fix some $S \subseteq V$ and notice that

$$\begin{aligned} \mathbb{E} [\text{cut}_{H'}(S)] &= \mathbb{E} \left[\sum_{e \in E} I_e \cdot \frac{1}{p_e} g_e(S \cap e) \right] = \sum_{e \in E} \frac{g_e(S \cap e)}{p_e} \cdot \mathbb{E} [I_e] \\ &= \sum_{e \in E} g_e(S \cap e) = \text{cut}_H(S). \end{aligned}$$

97:12 Cut Sparsification and Succinct Representation of Submodular Hypergraphs

Hence, the cut is preserved in expectation. We shall now prove that the value of the cut is concentrated around its expectation. Let $Q_S = \{e \in E : p_e \in (0, 1) \wedge g_e(S \cap e) > 0\}$ be the set of all hyperedges whose contribution to $\text{cut}_{H'}(S)$ is random. Furthermore, denote the maximum contribution of any such hyperedge to $\text{cut}_{H'}(S)$ by $b := \max_{e \in Q_S} p_e^{-1} g_e(S \cap e)$. By the Chernoff bound for bounded variables (Theorem A.1),

$$\Pr[\text{cut}_{H'}(S) \notin (1 \pm \epsilon) \cdot \text{cut}_H(S)] \leq 2 \cdot \exp\left(-\frac{\epsilon^2 \cdot \text{cut}_H(S)}{b}\right). \quad (6)$$

We first analyze the special case $S = V$. Observe that if $\text{cut}_H(V) = 0$ then the cut is preserved trivially. Otherwise, note that $p_e \geq M \rho'_e = \frac{M g_e(e)}{\sum_{f \in E} g_f(f)}$ and hence

$$b = \max_{e \in Q_V} \frac{g_e(e)}{p_e} \leq \max_{e \in Q_V} g_e(e) \frac{\sum_{f \in E} g_f(f)}{M g_e(e)} = \frac{\text{cut}_H(V)}{M}.$$

Plugging this into Equation (6), we find $\Pr[\text{cut}_{H'}(V) \notin (1 \pm \epsilon) \cdot \text{cut}_H(V)] \leq 2 \cdot \exp(-\epsilon^2 M)$. Now turning to the general case $S \subset V$, observe that by Corollary 2.2, $p_e \geq M g_e(S \cap e) / \text{cut}_H(S)$. Hence, we again obtain that

$$b \leq \max_{e \in E} g_e(S \cap e) \cdot \frac{\text{cut}_H(S)}{M g_e(S \cap e)} = \frac{\text{cut}_H(S)}{M}. \quad (7)$$

Plugging this back into our concentration bound, Equation (6), we get

$$\Pr[\text{cut}_{H'}(S) \notin (1 \pm \epsilon) \cdot \text{cut}_H(S)] \leq 2 \cdot \exp(-\epsilon^2 M).$$

Notice that this is the same probability as the case $S = V$. Setting $M := c \cdot \epsilon^{-2} n$ for large enough but fixed $c > 0$, we get that $\text{cut}_{H'}(S)$ approximates $\text{cut}_H(S)$ up to a $1 \pm \epsilon$ factor with probability at least $1 - 2 \exp(-cn)$. Applying a union bound over all $S \subseteq V$ we get that the sparsifier approximates all cuts simultaneously with probability at least $1 - 2 \exp(-cn) \cdot 2^n \geq 1 - 2 \exp(-n)$. This completes the construction of a quality $1 + \epsilon$ sparsifier for H with $O(\epsilon^{-2} n^3)$ hyperedges.

We now turn back to proving Claim 2.1.

Proof of Claim 2.1. Fix some $e \in E$ and $S \subset V$. For each directed minimum cut, let $P_e^{u \rightarrow v} := \arg \min_{S \subseteq V: S \cap \{u, v\} = \{u\}} g_e(S)$ be some set $S \subseteq V$ attaining the minimum cut value (breaking ties arbitrarily). We need to show that

$$\max_{u \in S, v \in \bar{S}} g_e(P_e^{u \rightarrow v}) \leq g_e(S \cap e) \leq \sum_{u \in S} \sum_{v \in \bar{S}} g_e(P_e^{u \rightarrow v}). \quad (8)$$

The lower bound is immediate because $g_e(P_e^{u \rightarrow v})$ is a minimizer over the cuts separating u from v . For the upper bound, since g_e is submodular and non-negative,

$$\forall A, B \subseteq e, \quad g_e(A) + g_e(B) \geq g_e(A \cap B) + g_e(A \cup B) \geq g_e(A \cap B),$$

and similarly, $g_e(A) + g_e(B) \geq g_e(A \cup B)$. Using these two inequalities and summing over all $v \in \bar{S}$ and $u \in S$, we get

$$\sum_{u \in S} \sum_{v \in \bar{S}} g_e(P_e^{u \rightarrow v}) \geq \sum_{u \in S} g_e\left(\bigcap_{v \in \bar{S}} P_e^{u \rightarrow v}\right) \geq g_e\left(\bigcup_{u \in S} \bigcap_{v \in \bar{S}} P_e^{u \rightarrow v}\right).$$

To conclude the proof we show that $S \cap e = \bigcup_{u \in S} \bigcap_{v \in \bar{S}} P_e^{u \rightarrow v}$. For all $u \in S \cap e$ we have $\{u\} \subseteq \bigcap_{v \in \bar{S}} P_e^{u \rightarrow v}$, therefore $S \cap e \subseteq \bigcup_{u \in S} \bigcap_{v \in \bar{S}} P_e^{u \rightarrow v}$. In addition, for all $u \in S$ we have $\bigcap_{v \in \bar{S}} P_e^{u \rightarrow v} \subseteq S \cap e$ if $u \in e$ and $P_e^{u \rightarrow v} = \emptyset$ otherwise, therefore $S \cap e = \bigcup_{u \in S} \bigcap_{v \in \bar{S}} P_e^{u \rightarrow v}$. We conclude that Equation (8) holds. \triangleleft

This completes the proof of Theorem 1.4. \blacktriangleleft

2.1 Monotone Submodular Hypergraphs

This section proves that every monotone submodular hypergraph admits a quality $(1 + \epsilon)$ -sparsifier of size $O(\epsilon^{-2}n^2)$.

► **Theorem 2.3.** *Every hypergraph with monotone splitting functions admits a quality $(1 + \epsilon)$ -sparsifier of size $O(\epsilon^{-2}n^2)$, which is a reweighted sub-hypergraph.*

The proof for the monotone case is similar to the general case. However, since monotone splitting functions are more structured it suffices to examine the importance of all the singleton cuts for each hyperedge. This results in smaller sampling probabilities and a better bound on the number of hyperedges in the sparsifier. The proof utilizes the following well known property of monotone submodular functions.

▷ **Claim 2.4.** Let $g_e : 2^V \rightarrow \mathbb{R}_+$ be a monotone submodular splitting function. Then

$$\forall S \subseteq V, \quad \max_{v \in S} g_e(\{v\} \cap e) \leq g_e(S \cap e) \leq \sum_{v \in S} g_e(\{v\} \cap e).$$

Proof. The lower bound holds as g_e is monotone. For the upper bound, since g_e is submodular and non-negative,

$$\sum_{v \in S} g_e(\{v\} \cap e) \geq g_e\left(\bigcup_{v \in S} \{v\} \cap e\right) = g_e(S \cap e). \quad \blacktriangleleft$$

Similarly to the general case, our over sampling probabilities are proportional to

$$\rho_e = \sum_{v \in V} \frac{g_e(\{v\} \cap e)}{\sum_{f \in E} g_f(\{v\} \cap f)}.$$

The following corollary shows that $\rho_e \geq \sigma_e$. This implies that sampling every $e \in E$ with probability proportional to ρ_e suffices to approximate the cuts of H , in the same manner as in the general case.

► **Corollary 2.5.** *For every $e \in E$ and $S \subseteq V$, we have $\rho_e \geq g_e(S \cap e) / \text{cut}_H(S)$.*

Proof. Observe that by Claim 2.4,

$$\frac{g_e(S \cap e)}{\text{cut}_H(S)} = \frac{g_e(S \cap e)}{\sum_{f \in E} g_f(S \cap f)} \leq \sum_{v \in S} \frac{g_e(\{v\} \cap e)}{\sum_{f \in E} g_f(\{v\} \cap f)} \leq \rho_e.$$

Notice that the first inequality is well-defined by the convention that if the denominator (and thus also the numerator) is zero then the fraction is zero. ◀

We now turn to proving Theorem 2.3

Proof of Theorem 2.3. To construct H' , sample each hyperedge with probability $p_e = \min(1, M \cdot \rho_e)$ for a suitable parameter $M = O(\epsilon^{-2}n)$. Then, reweigh every sampled hyperedge by factor p_e^{-1} . The proof that H' is with high probability a $(1 + \epsilon)$ -sparsifier is similar to the general case because $\rho_e \geq \sigma_e$, and we omit it.

To bound the number of hyperedges in the sparsifier, let I_e be an indicator for the event that the hyperedge e is sampled into H' . Then the expected number of sampled hyperedges is,

$$\begin{aligned} \mathbb{E} \left[\sum_{e \in E} I_e \right] &= \sum_{e \in E} p_e \leq M \sum_{e \in E} \sum_{v \in V} \frac{g_e(\{v\} \cap e)}{\sum_{f \in E} g_f(\{v\} \cap f)} \\ &\leq M \sum_{v \in V} \sum_{e \in E} \frac{g_e(\{v\} \cap e)}{\sum_{f \in E} g_f(\{v\} \cap f)} \leq Mn, \end{aligned}$$

where the second inequality is from changing the order of summation. Hence, by Markov's inequality we find that with high constant probability the size of the sparsifier is at most $O(Mn) = O(\epsilon^{-2}n^2)$. This concludes the proof. ◀

References

- 1 Ittai Abraham, David Durfee, Ioannis Koutis, Sebastian Krinninger, and Richard Peng. On fully dynamic graph sparsifiers. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS*, pages 335–344. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.44.
- 2 Alexandr Andoni, Jiecao Chen, Robert Krauthgamer, Bo Qin, David P. Woodruff, and Qin Zhang. On sketching quadratic forms. In *Innovations in Theoretical Computer Science, ITCS'16*, pages 311–319. ACM, 2016. doi:10.1145/2840728.2840753.
- 3 Alexandr Andoni, Anupam Gupta, and Robert Krauthgamer. Towards $(1 + \epsilon)$ -approximate flow sparsifiers. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 279–293. SIAM, 2014.
- 4 Sepehr Assadi and Sahil Singla. Improved truthful mechanisms for combinatorial auctions with submodular bidders. *SIGecom Exch.*, 18(1):19–27, 2020. doi:10.1145/3440959.3440964.
- 5 Nikhil Bansal, Ola Svensson, and Luca Trevisan. New notions and constructions of sparsification for graphs and hypergraphs. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019*, pages 910–928. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00059.
- 6 Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-Ramanujan sparsifiers. *SIAM Rev.*, 56(2):315–334, 2014. doi:10.1137/130949117.
- 7 András A. Benczúr and David R. Karger. Approximating s - t minimum cuts in $\tilde{O}(n^2)$ time. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 47–55. ACM, 1996. doi:10.1145/237814.237827.
- 8 Charles Carlson, Alexandra Kolla, Nikhil Srivastava, and Luca Trevisan. Optimal lower bounds for sketching graph cuts. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2565–2569, 2019. doi:10.1137/1.9781611975482.158.
- 9 Ruoxu Cen, Yu Cheng, Debmalya Panigrahi, and Kevin Sun. Sparsification of directed graphs via cut balance. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, volume 198 of *LIPICs*, pages 45:1–45:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ICALP.2021.45.
- 10 Yu Chen, Sanjeev Khanna, and Ansh Nagda. Near-linear size hypergraph cut sparsifiers. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 61–72. IEEE, 2020. doi:10.1109/FOCS46700.2020.00015.
- 11 Mahdi Cheraghchi, Adam R. Klivans, Pravesh Kothari, and Homin K. Lee. Submodular functions are noise stable. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*, pages 1586–1592. SIAM, 2012. doi:10.1137/1.9781611973099.126.
- 12 Michele Conforti and Gérard Cornuéjols. Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discret. Appl. Math.*, 7(3):251–274, 1984. doi:10.1016/0166-218X(84)90003-9.
- 13 Marcel Kenji de Carli Silva, Nicholas J. A. Harvey, and Cristiane M. Sato. Sparse sums of positive semidefinite matrices. *ACM Trans. Algorithms*, 12(1):9:1–9:17, 2016. doi:10.1145/2746241.
- 14 Nikhil R. Devanur, Shaddin Dughmi, Roy Schwartz, Ankit Sharma, and Mohit Singh. On the approximation of submodular functions. *CoRR*, abs/1304.4948, 2013. arXiv:1304.4948.
- 15 Shahar Dobzinski and Michael Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006*, pages 1064–1073. ACM Press, 2006. URL: <http://dl.acm.org/citation.cfm?id=1109557.1109675>.
- 16 Uriel Feige. On maximizing welfare when utility functions are subadditive. *SIAM J. Comput.*, 39(1):122–142, 2009. doi:10.1137/070680977.

- 17 Uriel Feige and Jan Vondrák. Approximation algorithms for allocation problems: Improving the factor of $1 - 1/e$. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, pages 667–676. IEEE Computer Society, 2006. doi:10.1109/FOCS.2006.14.
- 18 Vitaly Feldman and Pravesh Kothari. Learning coverage functions and private release of marginals. In *Proceedings of The 27th Conference on Learning Theory, COLT 2014*, volume 35 of *JMLR Workshop and Conference Proceedings*, pages 679–702. JMLR.org, 2014. URL: <http://proceedings.mlr.press/v35/feldman14a.html>.
- 19 Vitaly Feldman, Pravesh Kothari, and Jan Vondrák. Representation, approximation and learning of submodular functions using low-rank decision trees. In *COLT 2013 - The 26th Annual Conference on Learning Theory*, volume 30 of *JMLR Workshop and Conference Proceedings*, pages 711–740. JMLR.org, 2013. URL: <http://proceedings.mlr.press/v30/Feldman13.html>.
- 20 Vitaly Feldman and Jan Vondrák. Optimal bounds on approximation of submodular and XOS functions by juntas. *SIAM J. Comput.*, 45(3):1129–1170, 2016. doi:10.1137/140958207.
- 21 Wai-Shing Fung, Ramesh Hariharan, Nicholas J. A. Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. *SIAM J. Comput.*, 48(4):1196–1223, 2019. doi:10.1137/16M1091666.
- 22 Michel X. Goemans, Nicholas J. A. Harvey, Satoru Iwata, and Vahab S. Mirrokni. Approximating submodular functions everywhere. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009*, pages 535–544. SIAM, 2009. doi:10.1137/1.9781611973068.59.
- 23 Ryan Gomes and Andreas Krause. Budgeted nonparametric learning from data streams. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 391–398. Omnipress, 2010. URL: <https://icml.cc/Conferences/2010/papers/433.pdf>.
- 24 Anupam Gupta, Moritz Hardt, Aaron Roth, and Jonathan R. Ullman. Privately releasing conjunctions and the statistical query barrier. *SIAM J. Comput.*, 42(4):1494–1520, 2013. doi:10.1137/110857714.
- 25 Arun Jambulapati, James R. Lee, Yang P. Liu, and Aaron Sidford. Sparsifying sums of norms. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023*, pages 1953–1962. IEEE, 2023. doi:10.1109/FOCS57990.2023.00119.
- 26 Arun Jambulapati, Victor Reis, and Kevin Tian. Linear-sized sparsifiers via near-linear time discrepancy theory. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5169–5208. SIAM, 2024. doi:10.1137/1.9781611977912.186.
- 27 Michael Kapralov, Robert Krauthgamer, Jakab Tardos, and Yuichi Yoshida. Towards tight bounds for spectral sparsification of hypergraphs. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 598–611. ACM, 2021. doi:10.1145/3406325.3451061.
- 28 Sanjeev Khanna, Aaron Putterman, and Madhu Sudan. Code sparsification and its applications. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5145–5168. SIAM, 2024. doi:10.1137/1.9781611977912.185.
- 29 Dmitry Kogan and Robert Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015*, pages 367–376. ACM, 2015. doi:10.1145/2688073.2688093.
- 30 Jannik Kudla and Stanislav Zivný. Sparsification of monotone k -submodular functions of low curvature. *CoRR*, abs/2302.03143, 2023. doi:10.48550/arXiv.2302.03143.
- 31 Pan Li and Olga Milenkovic. Inhomogeneous hypergraph clustering with applications. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 2308–2318, 2017. URL: <https://proceedings.neurips.cc/paper/2017/hash/a50abba8132a77191791390c3eb19fe7-Abstract.html>.
- 32 Pan Li and Olga Milenkovic. Submodular hypergraphs: p-laplacians, cheeger inequalities and spectral clustering. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3020–3029. PMLR, 2018. URL: <http://proceedings.mlr.press/v80/li18e.html>.

97:16 Cut Sparsification and Succinct Representation of Submodular Hypergraphs

- 33 Hui Lin and Jeff A. Bilmes. A class of submodular functions for document summarization. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference 2011*, pages 510–520. The Association for Computer Linguistics, 2011. URL: <https://aclanthology.org/P11-1052/>.
- 34 Meng Liu, Nate Veldt, Haoyu Song, Pan Li, and David F. Gleich. Strongly local hypergraph diffusions for clustering and semi-supervised learning. In *WWW '21: The Web Conference 2021*, pages 2092–2103. ACM / IW3C2, 2021. doi:10.1145/3442381.3449887.
- 35 S Thomas McCormick. Submodular function minimization. *Handbooks in operations research and management science*, 12:321–391, 2005.
- 36 Kazusato Oko, Shinsaku Sakaue, and Shin-ichi Tanigawa. Nearly tight spectral sparsification of directed hypergraphs. In *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023*, volume 261 of *LIPICs*, pages 94:1–94:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.ICALP.2023.94.
- 37 Yosef Pogrow. Solving symmetric diagonally dominant linear systems in sublinear time (and some observations on graph sparsification). Master’s thesis, Weizmann Institute of Science, 2017. URL: https://www.wisdom.weizmann.ac.il/~robi/files/YosefPogrow-MScThesis-2017_12.pdf.
- 38 Kent Quanrud. Quotient sparsification for submodular functions. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5209–5248. SIAM, 2024. doi:10.1137/1.9781611977912.187.
- 39 Akbar Rafiey and Yuichi Yoshida. Sparsification of decomposable submodular functions. In *Thirty-Sixth AAAI Conference on Artificial Intelligence*, pages 10336–10344. AAAI Press, 2022. doi:10.1609/aaai.v36i9.21275.
- 40 Tasuku Soma and Yuichi Yoshida. Spectral sparsification of hypergraphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 2570–2581. SIAM, 2019. doi:10.1137/1.9781611975482.159.
- 41 Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011. doi:10.1137/080734029.
- 42 Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011. doi:10.1137/08074489X.
- 43 Sebastian Tschieschek, Rishabh K. Iyer, Haochen Wei, and Jeff A. Bilmes. Learning mixtures of submodular functions for image collection summarization. In *Advances in Neural Information Processing Systems 27 (NeurIPS 2014)*, pages 1413–1421, 2014. URL: <https://proceedings.neurips.cc/paper/2014/hash/a8e864d04c95572d1aece099af852d0a-Abstract.html>.
- 44 Nate Veldt, Austin R. Benson, and Jon M. Kleinberg. Minimizing localized ratio cut objectives in hypergraphs. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1708–1718. ACM, 2020. doi:10.1145/3394486.3403222.
- 45 Nate Veldt, Austin R. Benson, and Jon M. Kleinberg. Approximate decomposable submodular function minimization for cardinality-based components. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, pages 3744–3756, 2021. URL: <https://proceedings.neurips.cc/paper/2021/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html>.
- 46 Nate Veldt, Austin R. Benson, and Jon M. Kleinberg. Hypergraph cuts with general splitting functions. *SIAM Rev.*, 64(3):650–685, 2022. doi:10.1137/20m1321048.
- 47 Jan Vondrák. Submodularity and curvature: The optimal algorithm (combinatorial optimization and discrete algorithms). *RIMS Kokyuroku Bessatsu*, 23:253–266, 2010. URL: <http://hdl.handle.net/2433/177046>.
- 48 Yu Zhu, Boning Li, and Santiago Segarra. Hypergraph 1-spectral clustering with general submodular weights. In *56th Asilomar Conference on Signals, Systems, and Computers, ACSSC 2022*, pages 935–939. IEEE, 2022. doi:10.1109/IEEECONF56349.2022.10052065.

A Chernoff Bounds

We use the following version of the Chernoff bound throughout the paper.

► **Theorem A.1** (Chernoff bound for bounded random variables, Theorem 6.1 in [3]). *Let $X_1, \dots, X_m \geq 0$ be independent random variables such that either X_i is deterministic or $X_i \in [0, b]$. Let X denote their sum and $\mu = \mathbb{E}[X]$, then,*

$$\forall \delta > 0, \quad \Pr[X - \mu \geq \delta\mu] \leq 2 \cdot \exp\left(-\frac{\delta^2\mu}{(2 + \delta)b}\right).$$

Additionally,

$$\forall \delta \in [0, 1], \quad \Pr[|X - \mu| \geq \delta\mu] \leq 2 \cdot \exp\left(-\frac{\delta^2\mu}{3b}\right).$$