Spectral Sparsification of Hypergraphs

Tasuku Soma*
The University of Tokyo
tasuku_soma@mist.i.u-tokyo.ac.jp

Yuichi Yoshida†
National Institute of Informatics
yyoshida@nii.ac.jp

Abstract

For an undirected/directed hypergraph $G = (V, E)$, its Laplacian $L_G \colon \mathbb{R}^V \to \mathbb{R}^V$ is defined such that its "quadratic form" $\boldsymbol{x}^\top L_G(\boldsymbol{x})$ captures the cut information of G. In particular, $\mathbf{1}_S^\top L_G(\mathbf{1}_S)$ coincides with the cut size of $S \subseteq V$, where $\mathbf{1}_S \in \mathbb{R}^V$ is the characteristic vector of S.

A weighted subgraph H of a hypergraph G on a vertex set V is said to be an ϵ-spectral sparsifier of G if $(1-\epsilon)\boldsymbol{x}^\top L_H(\boldsymbol{x}) \leq \boldsymbol{x}^\top L_G(\boldsymbol{x}) \leq (1+\epsilon)\boldsymbol{x}^\top L_H(\boldsymbol{x})$ holds for every $\boldsymbol{x} \in \mathbb{R}^V$. In this paper, we present a polynomial-time algorithm that, given an undirected/directed hypergraph G on n vertices, constructs an ϵ-spectral sparsifier of G with $O(n^3 \log n/\epsilon^2)$ hyperedges/hyperarcs.

The proposed spectral sparsification can be used to improve the time and space complexities of algorithms for solving problems that involve the quadratic form, such as computing the eigenvalues of L_G, computing the effective resistance between a pair of vertices in G, semi-supervised learning based on L_G, and cut problems on G. In addition, our sparsification result implies that any nonnegative hypernetwork type submodular function can be concisely represented by a directed hypergraph of polynomial size, even if the original representation is of exponential size. Accordingly, we show that, for any distribution, we can properly and agnostically learn nonnegative hypernetwork type submodular functions with $O(n^4 \log(n/\epsilon)/\epsilon^4)$ samples.

1 Introduction

Let $G = (V, E, \boldsymbol{w})$ be a (weighted) graph, where $\boldsymbol{w} \in \mathbb{R}_+^E$ is a nonnegative weight function on edges. The *Laplacian* $L_G \in \mathbb{R}^{V \times V}$ of G is defined as

$$L_G(u,v) = \begin{cases} \sum_{e \in E : v \in e} \boldsymbol{w}(e) & \text{if } u = v, \\ -\boldsymbol{w}(e) & \text{if } e = \{u, v\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

A notable property of the Laplacian L_G is that its quadratic form can be written as

$$\boldsymbol{x}^\top L_G \boldsymbol{x} = \sum_{\{u,v\} \in E} \boldsymbol{w}(e)\big(\boldsymbol{x}(u) - \boldsymbol{x}(v)\big)^2.$$

This quadratic form captures many interesting properties of G. We say that an edge $e \in E$ is *cut* by a vertex set $S \subseteq V$ if $e \cap S \neq \emptyset, e \cap (V \setminus S) \neq \emptyset$. Then, for any vertex set $S \subseteq V$, the quantity $\mathbf{1}_S^\top L_G \mathbf{1}_S$ matches the *cut weight* of S, that is, the total weight of edges cut by S, where $\mathbf{1}_S \in \mathbb{R}^V$ is the characteristic vector of S. In addition, the quadratic form plays an important role in computing the eigenvalues of L_G because they can be obtained by minimizing the Rayleigh quotient $\mathcal{R}_G(\boldsymbol{x}) := \boldsymbol{x}^\top L_G \boldsymbol{x}/\|\boldsymbol{x}\|_2^2$ subject to some orthogonality constraints.

To reduce the time and space complexities of algorithms for solving problems that involve the quadratic form, it is desirable to approximate the quadratic form of a large input graph using that of a much smaller graph. For $\epsilon \in (0, 1)$, we say that a weighted subgraph H of a graph G on a vertex set V is an *ϵ-spectral sparsifier* of G if

$$(1-\epsilon)\boldsymbol{x}^\top L_H \boldsymbol{x} \leq \boldsymbol{x}^\top L_G \boldsymbol{x} \leq (1+\epsilon)\boldsymbol{x}^\top L_H \boldsymbol{x} \tag{1.1}$$

holds for every $\boldsymbol{x} \in \mathbb{R}^V$ [30]. If (1.1) holds (only) for characteristic vectors, H is called an *ϵ-cut sparsifier* because it preserves the cut weights. Note that an ϵ-spectral sparsifier is also an ϵ-cut sparsifier.

Cut sparsification and spectral sparsification of graphs have been studied intensively [1, 2, 17, 21, 22, 29], and it is known that any graph with n vertices can be spectrally sparsified with $\widetilde{O}(n/\epsilon)$ edges [17] or $O(n/\epsilon^2)$ edges [22], where $\widetilde{O}(\cdot)$ hides a polylogarithmic factor.

Spectral sparsifiers have numerous applications in theoretical computer science. First, as a spectral sparsifier H of G preserves the cut weights of G, it has been used to design efficient approximation algorithms related to cuts and flows [3, 4, 18, 24]. Second, we can speed up the computation of the eigenvalues of L_G because, as mentioned previously, they can be computed by minimizing the Rayleigh quotient $\mathcal{R}_G(\boldsymbol{x})$, which can be well approximated by $\mathcal{R}_H(\boldsymbol{x})$. Finally, spectral sparsification is a key technical tool for realizing nearly linear time algorithms for solving a Laplacian system, a linear system in the form of $L_G \boldsymbol{x} = \boldsymbol{b}$ for some vector

*Supported by CREST, JST.
†Supported by JSPS KAKENHI Grant Number JP17H04676

$\boldsymbol{b} \in \mathbb{R}^V$ [20, 31]. A fast Laplacian system solver yields to a fast symmetric diagonally dominant (SDD) system solver [31].

1.1 Our Contribution In this work, we consider spectral sparsification of hypergraphs, which could be undirected or directed. Recently, the notion of the Laplacian $L_G\colon \mathbb{R}^V \to \mathbb{R}^V$[1] for an undirected hypergraph $G = (V, E, \boldsymbol{w})$ was introduced [23, 32] such that its "quadratic form" $\boldsymbol{x}^\top L_G(\boldsymbol{x})$ satisfies

$$\boldsymbol{x}^\top L_G(\boldsymbol{x}) = \sum_{e \in E} \boldsymbol{w}(e) \max_{u,v \in e} \big(\boldsymbol{x}(u) - \boldsymbol{x}(v)\big)^2 \tag{1.2}$$

for every $\boldsymbol{x} \in \mathbb{R}^V$. We note that L_G is no longer a linear transformation. As with graphs, this quadratic form captures many properties of G. We say that a hyperedge $e \in E$ is *cut* by a vertex set $S \subseteq V$ if $e \cap S \neq \emptyset, e \cap (V \setminus S) \neq \emptyset$. Then, we can observe that $\mathbf{1}_S^\top L_G(\mathbf{1}_S)$ is equal to the cut weight of S. This quadratic form has been used to approximate the eigenvalues of L_G [32] as well as in semi-supervised learning [33].

As with graphs, for $\epsilon \in (0, 1)$, we say that a weighted subhypergraph H of an undirected hypergraph G on a vertex set V is an *ϵ-spectral sparsifier* of G if

$$(1-\epsilon)\boldsymbol{x}^\top L_H(\boldsymbol{x}) \leq \boldsymbol{x}^\top L_G(\boldsymbol{x}) \leq (1+\epsilon)\boldsymbol{x}^\top L_H(\boldsymbol{x}) \tag{1.3}$$

holds for every $\boldsymbol{x} \in \mathbb{R}^V$. If (1.3) holds (only) for characteristic vectors, H is called an *ϵ-cut sparsifier* of G. Although it is known that any undirected hypergraph on n vertices admits cut sparsifiers of $\widetilde{O}(n^2/\epsilon^2)$ hyperedges [19, 26], to the best our knowledge, no spectral sparsification has been studied thus far.

For an undirected hypergraph $G = (V, E, \boldsymbol{w})$, we define its *size* as $\mathrm{size}(G) = \sum_{e \in E} |e|$. In this work, we show that we can spectrally sparsify undirected hypergraphs with a polynomial number of hyperedges. Note that an undirected hypergraph on n vertices can have $\Omega(2^n)$ hyperedges.

THEOREM 1.1. *There exists a randomized algorithm that, given an undirected hypergraph $G = (V, E, \boldsymbol{w})$ and $\epsilon \in (0, 1)$, outputs an ϵ-spectral sparsifier H of G with $O(n^3 \log n/\epsilon^2)$ hyperedges with probability at least $1 - 1/n$ in $O(pn + m\log(1/\epsilon^2) + n^3 \log n/\epsilon^2)$ time, where $n = |V|$, $m = |E|$, and $p = \mathrm{size}(G)$.*

A directed hypergraph $G = (V, E, \boldsymbol{w})$ consists of a vertex set V, a set of hyperarcs E, and a weight function $\boldsymbol{w} \in \mathbb{R}_+^E$, where each hyperarc e is a pair (T_e, H_e) of (unnecessarily disjoint) sets of vertices, where $T_e, H_e \subseteq V$ are called the *tail* and *head*, respectively, of e. This notion was first introduced in [15], in which applications in propositional logic, dependency analysis in relational database, and traffic analysis were considered. A directed hypergraph coincides with an undirected hypergraph when $T_e = H_e$ for every $e \in E$ and a directed graph when $|T_e| = |H_e| = 1$ for every $e \in E$.

Using the framework of the submodular Laplacian [32], one can derive the Laplacian $L_G\colon \mathbb{R}^V \to \mathbb{R}^V$ for a directed hypergraph $G = (V, E, \boldsymbol{w})$, and its "quadratic form" $\boldsymbol{x}^\top L_G(\boldsymbol{x})$ can be written as

$$\boldsymbol{x}^\top L_G(\boldsymbol{x}) = \sum_{e=(T_e,H_e) \in E} \boldsymbol{w}(e) \max_{u \in T_e} \max_{v \in H_e} \big([\boldsymbol{x}(u) - \boldsymbol{x}(v)]^+\big)^2,$$

where $[x]^+ = \max\{x, 0\}$. We say that a hyperarc e is *cut* by a vertex set $S \subseteq V$ if $T_e \cap S \neq \emptyset$ and $H_e \cap (V \setminus S) \neq \emptyset$. As with the undirected case, the quadratic form can represent cut weights and has applications in semi-supervised learning [33]. We define ϵ-spectral and ϵ-cut sparsifiers in the same manner as in the undirected case.

For a directed hypergraph $G = (V, E, \boldsymbol{w})$, we define its *size* as as $\mathrm{size}(G) = \sum_{(T_e,H_e) \in E} (|T_e| + |H_e|)$. We show that we can spectrally sparsify directed hypergraphs with a polynomial number of hyperarcs.

THEOREM 1.2. *There exists a randomized algorithm that, given a weighted directed hypergraph $G = (V, E, \boldsymbol{w})$ and $\epsilon \in (0, 1)$, outputs an ϵ-spectral sparsifier H of G with $O(n^3 \log n/\epsilon^2)$ hyperarcs with probability at least $1 - 1/n$ in $O(pn + m\log(1/\epsilon^2) + n^3 \log n/\epsilon^2)$ time, where $n = |V|$, $m = |E|$, and $p = \mathrm{size}(G)$.*

We note that even cut sparsifiers were not known for directed hypergraphs. This result is interesting because directed graphs, which could have $\Omega(n^2)$ arcs, do not admit non-trivial cut/spectral sparsification of size $o(n^2)$ [8], whereas directed hypergraphs, which could have $O(2^{2n}) = O(4^n)$ hyperarcs, admit non-trivial cut/spectral sparsification of size $\widetilde{O}(n^3)$.

1.2 Applications Our sparsification results can be obviously used to improve the time and space complexities of algorithms for solving problems that involve the quadratic forms of hypergraph Laplacians, such as minimizing/maximizing cut weight subject to some constraints, computing eigenvalues, solving Laplacian systems, and semi-supervised learning.

Moreover, sparsification of directed hypergraphs is useful for obtaining a concise representation of a submodular function. A set function $f\colon 2^V \to \mathbb{R}$ is called *submodular* if $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$

[1] Precisely speaking, the range of L_G is $2^{\mathbb{R}^V}$, that is, a set in $\mathbb{R}^V$. However, we simply regard its range as $\mathbb{R}^V$ because (i) $L_G(\boldsymbol{x})$ is a single point almost everywhere and (ii) $\boldsymbol{x}^\top \boldsymbol{y}$ has the same value for any $\boldsymbol{y} \in L_G(\boldsymbol{x})$.

for every $S, T \subseteq V$. The cut function $\kappa_G \colon 2^V \to \mathbb{R}_+$ of a hypergraph G is a well-known example of submodular functions. Fujishige and Patkar [13] showed that any nonnegative submodular set function $f \colon 2^V \to \mathbb{R}_+$ such that $f(\emptyset) = f(V) = 0$ and for any $i = 3, 4, \ldots, n$ and $X \in \binom{V}{i}$,

$$f^{(i)}(X) := \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} f(Y) \leq 0$$

can be represented as a cut function of some directed hypergraph. Such a submodular function is said to be *nonnegative hypernetwork type*. We immediately obtain the following by Theorem 1.2:

COROLLARY 1.1. *For any nonnegative hypernetwork type submodular function $f \colon 2^V \to \mathbb{R}_+$ and any $\epsilon \in (0, 1)$, there exists a directed hypergraph $G = (V, E, \boldsymbol{w})$ with $O(n^3 \log n/\epsilon^3)$ hyperarcs for $n = |V|$ such that its cut function $\kappa_G \colon 2^V \to \mathbb{R}_+$ satisfies $(1 - \epsilon)\kappa_G(S) \leq f(S) \leq (1 + \epsilon)\kappa_G(S)$ for every $S \subseteq V$.*

We note that a class of nonnegative hypernetwork submodular functions is a fairly large class of submodular functions, since we need $\Omega(2^n)$ hyperarcs in general to exactly represent the original function [13]. Corollary 1.1 implies that a slight approximation allows us to reduce the number of hyperarcs to a polynomial number in n.

As an application of Corollary 1.1, we show that the class of nonnegative hypernetwork type submodular functions $f \colon 2^V \to [0, 1]$ is properly and agnostically learnable with a small number of samples:

COROLLARY 1.2. *Let $\mathcal{C}$ be the class of nonnegative hypernetwork type submodular functions $f \colon 2^V \to [0, 1]$ and let $\mathcal{D}$ be any distribution on $2^V \times \mathbb{R}$. Then, for any $\epsilon > 0$ and $\delta \in (0, 1)$, there exists an algorithm that, by drawing $O(n^4 \log(n/\epsilon)/\epsilon^4 + \log(1/\delta)/\epsilon^2)$ samples from $\mathcal{D}$ for $n = |V|$, outputs a submodular function $h \colon 2^V \to \mathbb{R}_+$ that can be evaluated in polynomial time in n, and*

$$\mathop{\mathbf{E}}_{(S,b)\sim\mathcal{D}} |h(S) - b| \leq \mathrm{opt} + \epsilon,$$

holds with probability at least $1 - \delta$, where $\mathrm{opt} = \min_{f \in \mathcal{C}} \mathop{\mathbf{E}}_{(S,b)\sim D} |f(S) - b|$.

This improves the previous best result [7], which requires $n^{O(1/\epsilon^2)} \log(1/\delta)$ samples, only works when the marginal distribution of $\mathcal{D}$ over 2^V is a product distribution, and is improper, that is, it may output a function that is not submodular.

A drawback of the algorithm presented in Corollary 1.2 is that it may require exponential time although the number of samples is a polynomial. By contrast, the algorithm presented in [7] only requires $n^{O(1/\epsilon^2)} \log(1/\delta)$ time. However, this is as expected, because agnostic learning of submodular functions $f \colon 2^V \to [0, 1]$ with $f(\emptyset) = f(V) = 0$ in $n^{o(1/\epsilon^{2/3})}$ time implies learning k-parities with noise in $n^{o(k)}$ time [11][2], which is a long-standing open problem in learning theory. Hence, Corollary 1.2 reveals a gap between the sample complexity and the time complexity for learning submodular functions.

1.3 Proof Ideas We now describe our proof ideas. For simplicity, let us assume that the input hypergraph $G = (V, E, \boldsymbol{w})$ is undirected and unweighted, that is, $\boldsymbol{w}(e) = 1$ for every $e \in E$. The arguments for the directed and weighted cases are more tedious but the ideas are similar.

First, we describe our construction of sparsifiers. For vertices $u, v \in V$, let $d_G(u, v)$ be the number of hyperedges including both u and v. For a hyperedge $e \in E$, let $p_e = 1/\min_{u,v\in e} d_G(u, v)$. Then, to construct an ϵ-spectral sparsifier H of G, for each hyperedge $e \in E$, we repeat the following process $K := O(n \log n/\epsilon^2)$ times. Add a copy of e with weight $\frac{1}{Kp_e}$ to H with probability p_e and do nothing for the complement probability $1 - p_e$. For each $e \in E$, the expected weight of copies of e in H is exactly one, and the expected number of hyperedges in H is $K \sum_{e\in E} p_e$.

The difficulty in the analysis arises because (1.2) involves a maximum operation, which makes it difficult to use linear algebraic tools. A key observation for resolving this issue is that (1.2) coincides with the quadratic form of the Laplacian associated with an undirected graph if we fix the ordering of elements in the vector $\boldsymbol{x} \in \mathbb{R}^V$. To see this, assume that $V = \{1, 2, \ldots, n\}$ and let π be a permutation of $[n]$ such that $\boldsymbol{x}(\pi_1) \geq \cdots \geq \boldsymbol{x}(\pi_n)$ (break ties arbitrary). Let G_π be the graph obtained by replacing each hyperedge $e \in E$ with an edge $\{s_e, t_e\}$, where $s_e = \mathrm{argmin}_{v\in e} \pi_v^{-1}$ and $t_e = \mathrm{argmax}_{v\in e} \pi_v^{-1}$. In other words, s_e and t_e are the first and last vertices of e in the ordering $\pi_1, \ldots, \pi_n$, respectively. Then, we have

$$\begin{aligned}
\boldsymbol{x}^\top L_G(\boldsymbol{x}) &= \sum_{e\in E} \max_{u,v\in e} \big(\boldsymbol{x}(u) - \boldsymbol{x}(v)\big)^2 \\
&= \sum_{e\in E} \big(\boldsymbol{x}(s_e) - \boldsymbol{x}(t_e)\big)^2 \\
&= \boldsymbol{x}^\top L_{G_\pi} \boldsymbol{x}.
\end{aligned}$$

[2] Although the original claim was about agnostic learning of *monotone* submodular functions, it was achieved first by showing the same claim for non-monotone submodular functions with $f(\emptyset) = f(V) = 0$ and using it.

This means that, if H_π is an ϵ-spectral sparsifier of G_π for *every* permutation π, then H is an ϵ-spectral sparsifier of G. It is known that, if we sample each edge e of G_π with a probability no less than a quantity called the effective resistance of e and repeat this process $O(\log n/\epsilon^2)$ times, we get an ϵ-spectral sparsifier of G_π with high probability [29]. Hence, a naive approach is sampling each hyperedge e with a probability no less than the maximum effective resistance of the corresponding edges of e in G_π over the choice of π and repeating this process $O(\log n/\epsilon^2)$ times. However, as discussed in Appendix A, this strategy may leave a hypergraph with an exponential number of hyperedges.

A crucial observation for addressing the above-mentioned issue is that, when constructing an ϵ-spectral sparsifier of G_π, we need to consider vectors only in $\mathbb{R}^V_\pi := \{\boldsymbol{x} \in \mathbb{R}^V \mid \boldsymbol{x}(\pi_1) \geq \cdots \geq \boldsymbol{x}(\pi_n)\}$, that is, we only need to guarantee that

$$(1-\epsilon)\boldsymbol{x}^\top L_{H_\pi}\boldsymbol{x} \leq \boldsymbol{x}^\top L_{G_\pi}\boldsymbol{x} \leq (1+\epsilon)\boldsymbol{x}^\top L_{H_\pi}\boldsymbol{x}$$

for any vector $\boldsymbol{x} \in \mathbb{R}^V_\pi$. In what follows, we only discuss how to guarantee $(1-\epsilon)\boldsymbol{x}^\top L_{H_\pi}\boldsymbol{x} \leq \boldsymbol{x}^\top L_{G_\pi}\boldsymbol{x}$ as the other direction is similar.

Suppose that we have constructed a hypergraph H based on $d_G(u,v)$ as mentioned above (that is, we set $p_e = 1/\min_{u,v\in e} d_G(u,v)$). To show that $(1-\epsilon)\boldsymbol{x}^\top L_{H_\pi}\boldsymbol{x} \leq \boldsymbol{x}^\top L_{G_\pi}\boldsymbol{x}$ for $\boldsymbol{x} \in \mathbb{R}^V_\pi$, we reparameterize $\boldsymbol{x}$ using the $(n-1)$-dimensional vector $\boldsymbol{y} = (\boldsymbol{x}(\pi_1) - \boldsymbol{x}(\pi_2), \ldots, \boldsymbol{x}(\pi_{n-1}) - \boldsymbol{x}(\pi_n))$. Let $Q_\pi \in \mathbb{R}^{(n-1)\times n}$ be the matrix such that $\boldsymbol{y} = Q_\pi\boldsymbol{x}$. Then, it is possible to represent $L_{G_\pi} = Q_\pi^\top B_\pi^\top B_\pi Q_\pi$ and $L_{H_\pi} = Q_\pi^\top B_\pi^\top W_H B_\pi Q_\pi$ for some matrices $B_\pi \in \mathbb{R}^{E\times(n-1)}$ and $W_H \in \mathbb{R}^{E\times E}$. Here, B_π is a $\{0,1\}$-valued matrix with each row having an interval of consecutive ones and W_H is a diagonal matrix whose (e,e)-th element is the total weight of copies of e in H. Then, it suffices to show that $(1-\epsilon)\boldsymbol{y}^\top B_\pi^\top W_H B_\pi \boldsymbol{y} \leq \boldsymbol{y}^\top B_\pi^\top B_\pi \boldsymbol{y}$ for every $\boldsymbol{y} \in \mathbb{R}^{n-1}_+$, that is, $B_\pi^\top B_\pi - (1-\epsilon)B_\pi^\top W_H B_\pi$ is *copositive*. To show that the matrix is copositive, we show by using a concentration bound that, as long as p_e is no less than some threshold p_e^π, we have $B_\pi^\top B_\pi - (1-\epsilon)B_\pi^\top W_H B_\pi \geq O$ with high probability, where $O \in \mathbb{R}^{(n-1)\times(n-1)}$ is the zero matrix. Then, we show that $\max_\pi p_e^\pi = 1/\min_{u,v\in e} d_G(u,v)$. This implies that, from our choice of p_e, the resulting hypergraph H acts as an ϵ-spectral sparsifier H_π of G_π for every permutation π with high probability, which means that H is an ϵ-spectral sparsifier of G with high probability. As we can show that $\sum_{e\in E} p_e = \sum_{e\in E} 1/\min_{u,v\in e} d_G(u,v) = O(n^2)$ holds, H has $O(n^2K) = O(n^3\log n/\epsilon^2)$ hyperedges on average.

1.4 Related Work In this section n, m, and p denote the number of vertices, number of edges, and size, respectively, of the input (hyper)graph.

The first general construction of cut sparsifiers for graphs is attributed to Benczúr and Karger [3]. They presented a polynomial-time algorithm that constructs an ϵ-cut sparsifier with $O(n\log n/\epsilon^2)$ edges. Their sparsifier is constructed by randomly sampling each edge with probability proportional to a parameter called *edge strength*. Fung *et al.* [14] showed that we can construct an ϵ-cut sparsifier with $O(n\log n/\epsilon^2)$ edges by sampling each edge with probability proportional to (an approximation to) the edge-connectivity of its end points, reducing the time complexity to $\widetilde{O}(n/\epsilon^2 + m)$.

Spielman and Teng [30] introduced the concept of a spectral sparsifier for graphs and proposed a construction of a spectral sparsifiers with $O(n\log^c n)$ edges for some constant $c > 0$. The number of edges was later improved to $O(n\log n/\epsilon^2)$ by sampling every edge with probability proportional to its effective resistance [29]. Since then, various constructions of spectral sparsifier have been proposed [1, 2, 21] and the current best methods use merely $O(n/\epsilon^2)$ edges and run in $\widetilde{O}(m/\epsilon^c)$ time for some $c > 1$ [22] or use $\widetilde{O}(n/\epsilon)$ edges and run in $\widetilde{O}(m)$ time [17].

It was implicitly shown in [26] that every undirected hypergraph admits an ϵ-cut sparsifier with $\widetilde{O}(n^2\log n/\epsilon^c)$ hyperedges for some $c > 1$. Kogan and Krauthgamer [19] generalized the method of Benczúr and Karger [3] and showed that any r-uniform hypergraph admits an ϵ-cut sparsifier of $O(n(r+\log n)/\epsilon^2)$ hyperedges. Chekuri and Xu [6], based on the idea of [4], improved the running time to $O(p\log^2 n\log p + nr(r+\log n)/\epsilon^2)$, although the resulting cut sparsifier has $O(nr(r+\log n)/\epsilon^2)$ hyperedges, which is r times greater than the size of [19]. Theorem 1.1 extends these results to spectral sparsifiers and Theorem 1.2 further extends them to directed hypergraphs.

Several variants of Laplacians for hypergraphs have been proposed in the literature besides that the one used in this work. In [28], the cut value of a set $S \subseteq V$ was defined as $\sum_{e\in E} \boldsymbol{w}(e)|S\cap e||e\setminus S|$. De Carli Silva *et al.* [9] considered a hypergraph Laplacian by extending this cut function and showed that every hypergraph admits a spectral sparsifier of $O(n/\epsilon^2)$ hyperedges in this sense. Although they also provided a construction of cut sparsifiers for r-uniform hypergraphs with $O(n/\epsilon)$ edges using our definition of cut, their guarantee is merely $\frac{4(r-1)}{r^2}\mathbf{1}_S L_H(\mathbf{1}_S) \leq \mathbf{1}_S L_G(\mathbf{1}_S) \leq \frac{(1+\epsilon)r^2}{4(r-1)}\mathbf{1}_S L_H(\mathbf{1}_S)$ for every $S \subseteq V$, which is significantly weaker than the guarantee of ϵ-cut sparsifiers.

Learning of submodular functions with additive er-

ror was first considered by Gupta *et al.* [16], who showed that we can learn $[0,1]$-valued submodular functions with $n^{O(1/\epsilon^2)}$ value queries and applied the result to private data release. Cheraghchi *et al.* [7] showed that submodular functions are noise stable, which leads to agnostic learning of $[0,1]$-valued submodular functions with $n^{O(1/\epsilon^2)}$ samples from a product distribution. Subsequently, Feldman *et al.* [11] showed that PAC learning of submodular functions can be done in $\mathrm{poly}(n)\cdot 2^{O(1/\epsilon^4)}$ time for any distribution and that it requires $2^{\Omega(1/\epsilon^{2/3})}$ samples (or even value queries). As we mentioned previously, they also presented evidence that agnostic learning of submodular functions requires $n^{\Omega(1/\epsilon^{2/3})}$ time. Raskhodnikova and Yaroslavtsev [27] considered learning submodular functions taking values in the range $\{0,1,\dots,k\}$. They built up on the approach of [16] to show that such submodular functions can be expressed as $2k$-DNF and then applied Mansour's algorithm for learning DNF [25] to obtain a $\mathrm{poly}(n)\cdot k^{O(k\log k/\epsilon)}$-time PAC learning algorithm using value queries.

1.5 Organization The rest of this paper is organized as follows. Section 2 introduces basic concepts for hypergraphs and technical tools. In Section 3, we relate the sparsification for vectors in $\mathbb{R}^V_\pi$ for a permutation π and the cone of copositive matrices. Section 4 describes spectral sparsification of *graphs* for vectors in $\mathbb{R}^V_\pi$. Section 5 describes and analyzes our spectral sparsification of hypergraphs. Finally, Section 6 discusses some applications of spectral sparsification of hypergraphs.

2 Preliminaries

For a positive integer $n \in \mathbb{N}$, we denote the set $\{1,2,\dots,n\}$ by $[n]$. For a vector $\boldsymbol{x} \in \mathbb{R}^n$, we define $\mathrm{diag}(\boldsymbol{x}) \in \mathbb{R}^{n\times n}$ as the diagonal matrix whose (i,i)-th entry is $\boldsymbol{x}(i)$ for every $i \in [n]$. For a probabilistic event X, let $[X] \in \{0,1\}$ be the indicator of X.

Let $G = (V, E, \boldsymbol{w})$ be a hypergraph. The *size* of G, denoted by $\mathrm{size}(G)$, is $\sum_{e\in E}|e|$ if G is undirected and is $\sum_{(T_e,H_e)\in E}|T_e| + |H_e|$ if G is directed. For a set of hyperedges/hyperarcs $F \subseteq E$, we denote the subgraph $(V, E\setminus F, \boldsymbol{w}|_{E\setminus F})$ by $G \setminus F$, where $\boldsymbol{w}|_{E\setminus F} \in \mathbb{R}_+^{E\setminus F}$ is the restriction of $\boldsymbol{w}$ to $E \setminus F$.

The Moore-Penrose inverse of a matrix A is denoted by $A^\dagger$. The sets of $n \times n$ symmetric matrices and positive semidefinite matrices are denoted by $\mathbb{S}^n$ and $\mathbb{S}^n_+$, respectively. A symmetric matrix $A \in \mathbb{S}^n$ is called *copositive* if $\boldsymbol{x}^\top A\boldsymbol{x} \geq 0$ for any $\boldsymbol{x} \in \mathbb{R}^n_+$. For two symmetric matrices $A, B \in \mathbb{S}^n$, we write $A \succeq B$ if $A - B$ is positive semidefinite. For a symmetric matrix $A \in \mathbb{S}^n$, $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ denote the smallest and largest eigenvalues of A, respectively.

A convex cone C is said to be *pointed* if it does not contain a line, or equivalently, $\boldsymbol{x} \in C$ and $-\boldsymbol{x} \in C$ implies $\boldsymbol{x} = \boldsymbol{0}$.

LEMMA 2.1. (CHERNOFF'S BOUND) *Let $X_1,\dots,X_n$ be independent random variables bounded by the interval $[0,R]$. Then, for $X = X_1 + \cdots + X_n$ and $\epsilon \in (0,1)$, we have*

$$\Pr\left[|X-\mu| \geq \epsilon\mu\right] \leq 2\exp\left(-\frac{\epsilon^2\mu}{3R}\right).$$

where $\mu = \mathbf{E}[X]$.

Let π be a permutation of $[n]$. Then, let $\mathbb{R}^n_\pi$ be the set $\{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x}(\pi_1) \geq \cdots \geq \boldsymbol{x}(\pi_n)\}$ and let $P_\pi \in \mathbb{R}^{n\times n}$ be the permutation matrix associated with π, that is,

$$P_\pi(i,j) = \begin{cases} 1 & \text{if } j = \pi_i, \\ 0 & \text{otherwise.} \end{cases}$$

3 Cone of $\mathbb{R}^\pi_+$-Copositive Matrices

For a matrix $A \in \mathbb{S}^n$ and a permutation π, we say that A is $\mathbb{R}^\pi_+$-*copositive* if $\boldsymbol{x}^\top A\boldsymbol{x} \geq 0$ for any $\boldsymbol{x} \in \mathbb{R}^n_\pi$. In this section, we characterize $\mathbb{R}^\pi_+$-copositive matrices $A \in \mathbb{S}^n$ with $A\boldsymbol{1} = \boldsymbol{0}$ using copositive matrices.

PROPOSITION 3.1. *Let π be a permutation of $[n]$. The set $\mathcal{C}$ of symmetric $\mathbb{R}^\pi_+$-copositive matrices is a proper cone, that is, a pointed, closed, and full-dimensional convex cone.*

Proof. The closedness and convexity are trivial. The full-dimensionality follows since $\mathcal{C}$ contains the positive semidefinite cone, which is full-dimensional. For the pointedness, assume that $A, -A \in \mathcal{C}$. This implies that $\boldsymbol{x}^\top A\boldsymbol{x} = 0$ for any $\boldsymbol{x} \in \mathbb{R}^n_\pi$. Since $\dim \mathbb{R}^n_\pi = n$, $A = O$.

For two symmetric matrices $A, B \in \mathbb{S}^n$ and a permutation π of $[n]$, we define the relation $A \succeq_\pi B$ as $\boldsymbol{x}^\top(A-B)\boldsymbol{x} \geq 0$ for every $\boldsymbol{x} \in \mathbb{R}^n_\pi$. Note that A is $\mathbb{R}^\pi_+$-copositive if and only if $A \succeq_\pi O$. It is well known that any proper cone $\mathcal{C}$ induces a partial order $\succeq_\mathcal{C}$ by defining $x \succeq_\mathcal{C} y$ as $x - y \in \mathcal{C}$ (see [5, Section 2.4]). The following is immediate from this fact.

PROPOSITION 3.2. *For any permutation π of $[n]$, the relation $\succeq_\pi$ is a partial order on $\mathbb{S}^n$.*

To characterize $\mathbb{R}^\pi_+$-copositive matrices, we introduce a matrix $J \in \mathbb{R}^{(n-1)\times n}$ defined as

$$J(i,j) = \begin{cases} 1 & \text{if } j = i \\ -1 & \text{if } j = i+1 \\ 0 & \text{otherwise,} \end{cases}$$

that is,

$$J = \begin{bmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{bmatrix}.$$

LEMMA 3.1. *Let $A \in \mathbb{S}^n$ be a matrix with $A\mathbf{1} = \mathbf{0}$ and π be a permutation of $[n]$. Then, A is $\mathbb{R}^\pi_+$-copositive if and only if A is of the form $P_\pi^\top J^\top C J P_\pi$ for some copositive matrix $C \in \mathbb{R}^{(n-1)\times(n-1)}$.*

Proof. For simplicity, we show the lemma for $\pi = \mathrm{id}$. The general case is obtained by permuting variables with P_π.

For the sufficiency, since for any $\boldsymbol{x} \in \mathbb{R}^n_\pi$, $\boldsymbol{x}' := J\boldsymbol{x} \in \mathbb{R}^{n-1}_+$, we obtain $\boldsymbol{x}^\top J^\top C J \boldsymbol{x} = \boldsymbol{x}' C \boldsymbol{x}' \geq 0$.

For the necessity, let us consider an upper triangular matrix $E \in \mathbb{R}^{n\times n}$ defined as

$$E = \begin{bmatrix} 1 & \cdots & 1 \\ & \ddots & \vdots \\ & & 1 \end{bmatrix}.$$

Then E is a bijection from $\mathbb{R}^n_+$ to $\mathbb{R}^n_\pi$. For $\boldsymbol{x} \in \mathbb{R}^n_\pi$, defining $\boldsymbol{y} \in \mathbb{R}^n_+$ such that $\boldsymbol{x} = E\boldsymbol{y}$, we obtain $\boldsymbol{y}E^\top A E \boldsymbol{y} = \boldsymbol{x} A \boldsymbol{x} \geq 0$. Therefore $E^\top A E$ is copositive. Since $A\mathbf{1} = \mathbf{0}$ and the last column of E is $\mathbf{1}$, we have

$$E^\top A E = \begin{bmatrix} C & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix}$$

for some $(n-1)\times(n-1)$ copositive matrix C. From $E^{-1} = \begin{bmatrix} J \\ \boldsymbol{e}_n^\top \end{bmatrix}$, we conclude that

$$A = \begin{bmatrix} J^\top & \boldsymbol{e}_n \end{bmatrix} \begin{bmatrix} C & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} \begin{bmatrix} J \\ \boldsymbol{e}_n^\top \end{bmatrix} = J^\top C J.$$

4 Spectral Sparsification of Graphs for Vectors in $\mathbb{R}^n_\pi$

Let G be a (weighted) graph on the vertex set $V := [n]$ and π be a permutation of $[n]$. We say that a weighted subgraph H of G is an *ϵ-spectral sparsifier of G for vectors in $\mathbb{R}^n_\pi$* if $(1-\epsilon)\boldsymbol{x}^\top L_G \boldsymbol{x} \leq \boldsymbol{x}^\top L_H \boldsymbol{x} \leq (1+\epsilon)\boldsymbol{x}^\top L_G \boldsymbol{x}$ holds for every $\boldsymbol{x} \in \mathbb{R}^n_\pi$, or more simply, $(1-\epsilon)L_H \preceq_\pi L_G \preceq_\pi (1-\epsilon)L_H$ holds. In this section, we present a randomized algorithm that constructs spectral sparsifiers for vectors in $\mathbb{R}^n_\pi$.

Our algorithm is a very simple sampling algorithm (Algorithm 1). Given a graph $G = (V, E, \boldsymbol{w}_G)$ and edge sampling probabilities $\{p_e\}_{e\in E}$, we repeat the following process $K := \Theta(\log(n/\delta)/\epsilon^2)$ times: For each edge $e \in E$, we increase the weight $\boldsymbol{w}_H(e)$ by $\boldsymbol{w}_G(e)/(Kp_e)$ with probability p_e. Then, we output the graph $H = (V, E, \boldsymbol{w}_H)$. Clearly, the expected number of edges with non-zero weights in the output graph is $O\big(K\sum_{e\in E} p_e\big) = O\big(\sum_{e\in E} p_e \log(n/\delta)/\epsilon^2\big)$, which is small if $\sum_{e\in E} p_e$ is small.

Algorithm 1

Input: A graph $G = (V, E, \boldsymbol{w}_G)$, a permutation π, $\epsilon, \delta \in (0,1)$, and $\{p_e\}_{e\in E}$.

1: $K \leftarrow \Theta(\log(n/\delta)/\epsilon^2)$.
2: $\boldsymbol{w}_H \leftarrow \mathbf{0}$.
3: Let $(Z_{k,e})_{k\in[K],e\in E}$ be mutually independent random variables in $\{0,1\}$ with $\mathbf{E}[Z_{k,e}] = p_e$.
4: **for** $k \leftarrow 1$ to K **do**
5: **for** each $e \in E$ **do**
6: Increase $\boldsymbol{w}_H(e)$ by $Z_{k,e}\boldsymbol{w}_G(e)/(Kp_e)$.
7: **return** a graph $H = (V, E, \boldsymbol{w}_H)$.

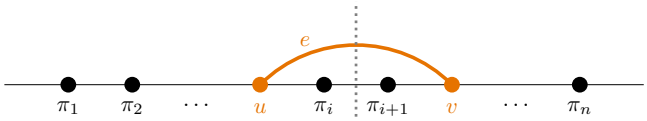

Figure 1: An edge crossing i in the order induced by π.

We will show that Algorithm 1 produces a spectral sparsifier with high probability if the sampling probabilities are higher than some thresholds. To describe the thresholds, we introduce some definitions. For $i \in [n-1]$, we say that an edge $e = \{u, v\} \in E$ *crosses i in the ordering induced by π* if $\min\{\pi_u^{-1}, \pi_v^{-1}\} \leq i < \max\{\pi_u^{-1}, \pi_v^{-1}\}$, and we denote it by $i \in_\pi e$. In words, when we align the vertices of G as $\pi_1, \ldots, \pi_n$ on a line in this order, the edge $\{u, v\}$ crosses the bisector between π_i and π_{i+1} (see Figure 1). For $i, j \in [n-1]$, let $d_{G,\pi}(i, j)$ be the total weight of edges that cross both i and j in π. Note that $d_{G,\pi}(i, j) = d_{G,\pi}(j, i)$.

In the rest of this section, we prove the following:

THEOREM 4.1. *Suppose that*

$$p_e \geq \frac{\boldsymbol{w}_G(e)}{\min_{i,j\in_\pi e} d_{G,\pi}(i,j)}$$

holds for every $e \in E$. Then, the graph H produced by Algorithm 1 is an ϵ-spectral sparsifier for vectors in $\mathbb{R}^n_\pi$ with probability at least $1-\delta$.

Our goal is to show that $(1-\epsilon)L_H \preceq_\pi L_G \preceq_\pi (1+\epsilon)L_H$ under the assumption of Theorem 4.1. To this end, we rephrase this condition using Lemma 3.1. Let $W_G, W_H \in \mathbb{R}^{E\times E}$ be the diagonal matrices whose (e,e)-th entries are $\boldsymbol{w}_G(e)$ and $\boldsymbol{w}_H(e)$, respectively, for

each $e \in E$. Then, we can write $L_G = B_G^\top W_G B_G$ and $L_H = B_G^\top W_H B_G$, where $B_G \in \mathbb{R}^{E \times V}$ is the incidence matrix of G. The following is a simple but important observation:

PROPOSITION 4.1. *We can write $B_G = B_\pi J P_\pi$, where $B_\pi \in \mathbb{R}^{E \times (n-1)}$ is a matrix defined as*

$$B_\pi(e,i) = \begin{cases} 1 & \text{if } i \in_\pi e, \\ 0 & \text{otherwise.} \end{cases}$$

Note that each row of B_π has consecutive ones. Then, Lemma 3.1 implies that $(1-\epsilon)L_H \preceq_\pi L_G \preceq_\pi (1+\epsilon)L_H$ holds if and only if $(1+\epsilon)B_\pi^\top W_H B_\pi - B_\pi^\top W_G B_\pi$ and $B_\pi^\top W_G B_\pi - (1-\epsilon)B_\pi^\top W_H B_\pi$ are copositive. The elements of these matrices can be written as follows:

LEMMA 4.1. *For $i,j \in [n-1]$, we have $(B_\pi^\top W_G B_\pi)(i,j) = d_{G,\pi}(i,j)$ and $(B_\pi^\top W_H B_\pi)(i,j) = d_{H,\pi}(i,j)$.*

Proof. We have

$$\begin{aligned}(B_\pi^\top W_G B_\pi)(i,j) &= \sum_{e \in E} \boldsymbol{w}_G(e)[i \in_\pi e \wedge j \in_\pi e] \\ &= d_{G,\pi}(i,j).\end{aligned}$$

The same argument holds for $(B_\pi^\top W_H B_\pi)(i,j)$.

Proof. [Proof of Theorem 4.1] To show the copositivity of $(1+\epsilon)B_\pi^\top W_H B_\pi - B_\pi^\top W_G B_\pi$ and $B_\pi^\top W_G B_\pi - (1-\epsilon)B_\pi^\top W_H B_\pi$, it suffices to show that they are non-negative matrices with probability at least $1-\delta$. To this end, we show that, for every $i,j \in [n-1]$, $(1-\epsilon)d_{H,\pi}(i,j) \le d_{G,\pi}(i,j) \le (1+\epsilon)d_{G,\pi}(i,j)$ with probability at least $1 - \delta/\binom{n}{2}$. Then, a union bound over the choice of $i,j \in [n-1]$ and Lemma 4.1 gives the desired result.

Fix $i,j \in [n-1]$. Note that $d_{H,\pi}(i,j) = \sum_{k,e} X_{k,e}$, where

$$X_{k,e} = \frac{Z_{k,e}\boldsymbol{w}_G(e)}{K p_e}[i,j \in_\pi e].$$

Then, we have

$$\begin{aligned}\mathbf{E}\big[d_{H,\pi}(i,j)\big] &= K \sum_{e \in E} \frac{\mathbf{E}[Z_{k,e}]\boldsymbol{w}_G(e)}{K p_e}[i,j \in_\pi e] \\ &= \sum_{e \in E} \boldsymbol{w}_G(e)[i,j \in_\pi e] = d_{G,\pi}(i,j).\end{aligned}$$

In addition, from the assumption on $\{p_e\}_{e \in E}$, we have

$$X_{k,e} \le \frac{\min_{i',j' \in_\pi e} d_{G,\pi}(i',j')}{K}[i,j \in_\pi e] \le \frac{d_{G,\pi}(i,j)}{K}.$$

Algorithm 2

Input: An undirected/directed hypergraph $G = (V, E, \boldsymbol{w}_G)$, $\epsilon, \delta \in (0,1)$, and $\{p_e\}_{e \in E}$.

1: $K \leftarrow \Theta(\log(n!/\delta)/\epsilon^2)$, where $n = |V|$.
2: $\boldsymbol{w}_H \leftarrow \mathbf{0}$.
3: Let $(Z_{k,e})_{k \in [K], e \in E}$ be mutually independent random variables in $\{0,1\}$ with $\mathbf{E}[Z_{k,e}] = p_e$.
4: **for** $k \leftarrow 1$ to K **do**
5: **for** each $e \in E$ **do**
6: Increase $\boldsymbol{w}_H(e)$ by $Z_{k,e}\boldsymbol{w}_G(e)/(Kp_e)$.
7: **return** a hypergraph $H = (V, E, \boldsymbol{w}_H)$.

By Chernoff's bound (Lemma 2.1), we have

$$\begin{aligned}\Pr\Big[|d_{H,\pi}(i,j) - d_{G,\pi}(i,j)| \ge \epsilon d_{G,\pi}(i,j)\Big] &\le 2\exp\Big(-\frac{\epsilon^2 K}{3}\Big) \\ &\le \frac{\delta}{\binom{n}{2}}.\end{aligned}$$

by setting the hidden constant in K to be sufficiently large.

5 Spectral Sparsification of Hypergraphs

In this section, we prove Theorems 1.1 and 1.2.

Our algorithm, given in Algorithm 2, is almost identical to Algorithm 1 except that the input and the output are now hypergraphs and that the error probability δ is divided by $n!$, and it works for both the undirected and directed cases by changing the sampling probabilities, as discussed in Sections 5.1 and 5.2, respectively.

5.1 Undirected Hypergraphs For a hypergraph $G = (V, E, \boldsymbol{w}_G)$ and two vertices $u, v \in V$, we define $d_G(u,v)$ as the total weight of hyperedges that include both u and v. Note that $d_G(v,u) = d_G(u,v)$. The following lemma gives a condition on the sampling probability of the resulting hypergraph being a spectral sparsifier.

LEMMA 5.1. *Suppose that*

$$p_e \ge \frac{\boldsymbol{w}_G(e)}{\min_{u,v \in e} d_G(u,v)}$$

holds for every $e \in E$. Then, Algorithm 2 produces an ϵ-spectral sparsifier of G with probability at least $1-\delta$.

Proof. For a permutation π, let G_π be the graph obtained from G by replacing each hyperedge $e \in E$ with an edge $e_\pi := \{s_e, t_e\}$, where $s_e = \operatorname{argmin}_{v \in e} \pi_v^{-1}$ and $t_e = \operatorname{argmax}_{v \in e} \pi_v^{-1}$. Note that, for every $\boldsymbol{x} \in \mathbb{R}_\pi^V$,

we have

$$\begin{aligned}\boldsymbol{x}^\top L_G(\boldsymbol{x}) &= \sum_{e\in E} \boldsymbol{w}_G(e) \max_{u,v\in e} \big(\boldsymbol{x}(u)-\boldsymbol{x}(v)\big)^2 \\ &= \sum_{e\in E} \boldsymbol{w}_G(e) \big(\boldsymbol{x}(s_e)-\boldsymbol{x}(t_e)\big)^2 \\ &= \boldsymbol{x}^\top L_{G_\pi} \boldsymbol{x}.\end{aligned}$$

For any $e \in E$, we have

$$\begin{aligned}\min_\pi \min_{i,j\in_\pi e} d_{G_\pi,\pi}(i,j) &= \min_{\pi:1,n-1\in_\pi e} d_{G_\pi,\pi}(1,n-1) \\ &= \min_{u,v\in e} d_G(u,v).\end{aligned}$$

(See Section 4 for the definition of $d_{G_\pi,\pi}(i,j)$.) Then, the claim holds by Theorem 4.1 and a union bound over the choice of π.

The following is useful for providing an upper bound on the number of hyperedges in H.

LEMMA 5.2. *We have*

$$\sum_{e\in E} \frac{\boldsymbol{w}_G(e)}{\min_{u,v\in e} d_G(u,v)} = O(n^2),$$

where $n = |V|$.

Proof. Let $\{u_1,v_1\},\dots,\{u_m,v_m\}$ be a sequence of (unordered) pairs of vertices in increasing order of $d_G(u_i,v_i)$, where $m = \binom{n}{2}$. Then, consider the following iterative algorithm with m steps. At step i, for each (remaining) hyperedge e including both u_i and v_i, we assign a cost of $\boldsymbol{w}_G(e)/d_G(u_i,v_i)$ to e and delete e.

The cost assigned to a hyperedge e is exactly $\boldsymbol{w}_G(e)/\min_{u,v\in e} d_G(u,v)$; hence we want to upper-bound the total cost assigned to the hyperedges. This can be bounded by $O(n^2)$ because the total cost assigned to the hyperedges at step i is at most 1 and the number of steps is $m = O(n^2)$.

We note that the bound $O(n^2)$ in Lemma 5.2 is tight because an unweighted complete graph satisfies $\sum_{e\in E} \boldsymbol{w}_G(e)/\min_{u,v\in e} d_G(u,v) = \sum_{e\in E} 1 = \binom{n}{2} = \Omega(n^2)$.

Proof. [Proof of Theorem 1.1] Our algorithm first calculates $d_G(u,v)$ for each (unordered) pair $\{u,v\} \in \binom{V}{2}$, and then calls Algorithm 2 with $p_e = \boldsymbol{w}_G(e)/\min_{u,v\in e} d_G(u,v)$ and $\delta = 1/2n$. By Lemma 5.1, the output is an ϵ-spectral sparsifier with probability at least $1-1/2n$ by setting the hidden constant in K to be sufficiently large. The expected number of hyperedges in the output is $O(K\sum_{e\in E} p_e) = O(n^3 \log n/\epsilon^2)$ by Lemma 5.2. Then, our algorithm outputs an ϵ-spectral sparsifier of $O(n^3 \log n/\epsilon^2)$ hyperedges with probability at least $1-1/n$.

Now, we analyze the time complexity. To compute $d_G(u,v)$'s, we introduce a counter c_{uv} initialized to be zero for each unordered pair $\{u,v\}$ of vertices. Then for each hyperedge $e \in E$, we increase the counter c_{uv} by $\boldsymbol{w}_G(e)$ for every unordered pair $\{u,v\} \subseteq e$. The time complexity for this part is $\sum_{e\in E} |e|^2 \le \sum_{e\in E} |e| n = pn$. To efficiently simulate the process from Line 4 to Line 6 in Algorithm 2, for each hyperedge $e \in E$, we set $\boldsymbol{w}_H(e)$ to be $\frac{\boldsymbol{w}_G(e) X_e}{K p_e}$, where X_e is sampled from the binomial distribution with a success probability p_e and the number of trials K, which can be done in $O(\log K) = O(\log(n \log n/\epsilon^2))$ time [10]. Hence, the total time complexity is $O(pn + m\log(n\log n/\epsilon^2) + n^3 \log n/\epsilon^2) = O(pn + m\log(1/\epsilon^2) + n^3 \log n/\epsilon^2)$.

5.2 Directed Hypergraphs The analysis for directed hypergraphs is almost identical to that for undirected hypergraphs. For a directed hypergraph $G = (V,E,\boldsymbol{w})$ and two vertices $u,v \in V$, we define $d_G(u,v)$ as the total weight of hyperarcs $e = (T_e,H_e)$ with $u \in T_e$ and $v \in H_e$. Note that, in this case, $d_G(u,v) \neq d_G(v,u)$ in general. The following lemma is analogous to Lemma 5.1.

LEMMA 5.3. *Suppose that*

$$p_e \ge \frac{\boldsymbol{w}_G(e)}{\min_{u\in T_e} \min_{v\in H_e} d_G(u,v)}$$

holds for every $e = (T_e,H_e) \in E$. *Then, Algorithm 2 produces an* ϵ*-spectral sparsifier of* G *with probability at least* $1-\delta$.

Proof. For a permutation π, let G_π be the undirected graph obtained from G by replacing each hyperarc $e = (T_e,H_e) \in E$ with an (undirected) edge $e_\pi := \{s,t\}$, where $s_e = \mathrm{argmin}_{u\in T_e} \pi_u^{-1}$ and $t_e = \mathrm{argmax}_{v\in H_e} \pi_v^{-1}$ if $\pi_{s_e}^{-1} < \pi_{t_e}^{-1}$, and deleting it otherwise. Note that, for every $\boldsymbol{x} \in \mathbb{R}_\pi^V$, we have

$$\begin{aligned}\boldsymbol{x}^\top L_G(\boldsymbol{x}) &= \sum_{e\in E} \boldsymbol{w}_G(e) \max_{u\in T_e} \max_{v\in H_e} \big([\boldsymbol{x}(u)-\boldsymbol{x}(v)]^+\big)^2 \\ &= \sum_{e\in E:\pi_{s_e}^{-1}<\pi_{t_e}^{-1}} \boldsymbol{w}_G(e) \big(\boldsymbol{x}(s_e)-\boldsymbol{x}(t_e)\big)^2 = \boldsymbol{x}^\top L_{G_\pi} \boldsymbol{x}.\end{aligned}$$

For any $e \in E$, we have

$$\begin{aligned}\min_\pi \min_{i,j\in_\pi e} d_{G_\pi,\pi}(i,j) &= \min_{\pi:1,n-1\in_\pi e} d_{G_\pi,\pi}(1,n-1) \\ &= \min_{u,v\in e} d_G(u,v).\end{aligned}$$

Then, the claim holds by Theorem 4.1 and a union bound over the choice of π.

The following lemma is analogous to Lemma 5.2.

LEMMA 5.4. *We have*

$$\sum_{e\in E} \frac{\boldsymbol{w}_G(e)}{\min_{u\in T_e}\min_{v\in H_e} d_G(u,v)} = O(n^2).$$

Proof. Let $(u_1,v_1),\dots,(u_m,v_m)$ be a sequence of (ordered) pairs of vertices in increasing order of $d_G(u_i,v_i)$, where $m = n(n-1)$. Then, consider the following iterative algorithm with m steps. At step i, for each (remaining) hyperarc $e = (T_e, H_e) \in E$ with $u_i \in T_e$ and $v_i \in H_e$, we assign a cost of $\boldsymbol{w}_G(e)/d_G(u_i,v_i)$ to e and delete e.

The cost assigned to a hyperarc $e = (T_e, H_e)$ is exactly $\boldsymbol{w}_G(e)/\min_{u\in T_e}\min_{v\in H_e} d_G(u,v)$; hence we want to upper-bound the total cost assigned to the hyperarcs. This can be bounded by $O(n^2)$ because the total cost assigned to the hyperarcs at step i is at most 1 and the number of steps is $m = O(n^2)$.

The proof of Theorem 1.2 is identical to that of Theorem 1.1, where we use Lemmas 5.3 and 5.4 instead of Lemmas 5.1 and 5.2.

6 Applications

In this section, we discuss applications of spectral sparsification of hypergraphs.

6.1 Agnostic Learning of Submodular Functions We now discuss an application of Corollary 1.1 to agnostic learning.

We first define agnostic learning for the case in which the domain is 2^V. Let $\mathcal{D}$ be an arbitrary distribution on $2^V \times [0,1]$. Let $\mathcal{C}$ be a class of $[0,1]$-valued functions on 2^V. Define the *error* of $f\colon 2^V \to [0,1]$ and the *optimal error* of $\mathcal{C}$ as

$$\mathrm{err}(f) = \mathop{\mathbf{E}}_{(S,b)\sim\mathcal{D}} |f(S) - b|, \quad \mathrm{opt} = \min_{f\in\mathcal{C}} \mathrm{err}(f),$$

respectively. Roughly speaking, the objective of agnostic learning of a concept class $\mathcal{C}$ is to find a hypothesis with an error not much larger than the optimal error by drawing a small number of samples from $\mathcal{D}$. Formally, we define agnostic learning as follows:

DEFINITION 6.1. (AGNOSTIC LEARNING) *A concept class $\mathcal{C}$ is said to be* agnostically learnable *if, for any $\epsilon > 0$ and $\delta \in (0,1)$, there exists an algorithm that, given a sampling oracle from $\mathcal{D}$, outputs a hypothesis $h\colon 2^V \to [0,1]$ with probability at least $1-\delta$ such that $\mathrm{err}(h) \le \mathrm{opt} + \epsilon$.*

The following is an easy consequence of Chernoff's bound and the union bound.

LEMMA 6.1. *A concept class $\mathcal{C}$ of $[0,1]$-valued functions is agnostically learnable with $O\big(\log(|\mathcal{C}|/\delta)/\epsilon^2\big)$ samples.*

Proof. [Proof of Corollary 1.2] Let $f\colon 2^V \to \mathbb{R}$ be a nonnegative hypernetwork type submodular function. We first show that f can be well approximated by a sparse directed hypergraph with small weights. Let $G = (V, E_G, \boldsymbol{w}_G)$ be a directed hypergraph such that $f(S) = \kappa_G(S)$ for every $S \subseteq V$ for the cut function $\kappa_G\colon 2^V \to \mathbb{R}_+$ of G, whose existence is guaranteed by Corollary 1.1. From the proof of Corollary 1.1 in [13], we can observe that $0 \le \boldsymbol{w}_G(e) \le \sum_{v\in V} f(v) \le n$. Let $M = O(n^3 \log n/\epsilon^2)$ be the upper bound on the expected number of hyperarcs in Theorem 1.2. By Theorem 1.2, there exists a directed hypergraph $H = (V, E_H, \boldsymbol{w}_H)$ of at most M hyperarcs such that $(1-\epsilon)\kappa_H(S) \le \kappa_G(S) \le (1+\epsilon)\kappa_H(S)$ for every $S \subseteq V$. Moreover, from the construction of H (Algorithm 2), we know that $E_H \subseteq E_G$ and

$$\boldsymbol{w}_H(e) \le \frac{\boldsymbol{w}_G(e)}{p_e} \le n \cdot n^2 = n^3$$

for every $e \in E_H$, where in the second inequality we used $\boldsymbol{w}_G(e) \le n$ and Lemma 5.4.

Let $\mathcal{C}$ be the class of nonegative hypernetwork type submodular functions $f\colon 2^V \to [0,1]$ and let $\mathcal{C}'$ be the class of cut functions of directed hypergraphs with at most M hyperarcs and each arc weight being a multiple of ϵ/M. Then, for every $f \in \mathcal{C}$, there exists $g \in \mathcal{C}'$ such that $|f(S) - g(S)| \le \epsilon$ for every $S \subseteq V$. Hence, it suffices to show that $\mathcal{C}'$ is agnostically learnable. Note that we have

$$|\mathcal{C}'| \le O\left(\sum_{k=0}^{M} \binom{2^{2n}}{k} \left(\frac{Mn^3}{\epsilon}\right)^k\right),$$

which implies that

$$\begin{aligned}\log|\mathcal{C}'| &= O\left(M\log 2^{2n} + M\log\frac{Mn^3}{\epsilon}\right)\\ &= O\left(\frac{n^4}{\epsilon^2}\log\frac{n}{\epsilon}\right).\end{aligned}$$

The claim follows by Lemma 6.1.

6.2 Effective Resistance of Hypergraphs As with graphs, we can define the effective resistance between a pair of vertices in a hypergraph. For $v \in V$, let $\boldsymbol{e}_v \in \mathbb{R}^V$ be the unit vector with $\boldsymbol{e}_v(v) = 1$ and $\boldsymbol{e}_v(w) = 0$ for $w \ne v$.

DEFINITION 6.2. (EFFECTIVE RESISTANCE [12]) *Let $G = (V, E, \boldsymbol{w})$ be an undirected/directed hypergraph.*

For distinct vertices $s, t \in V$, the effective resistance *$R_G(s,t)$ between s and t is the maximum value of*

$$2(\boldsymbol{e}_s - \boldsymbol{e}_t)^\top \boldsymbol{x} - \boldsymbol{x}^\top L_G(\boldsymbol{x}) \tag{6.4}$$

for $\boldsymbol{x} \in \mathbb{R}^V$.

The effective resistance has the following physics interpretation. Let us regard a hypergraph as an electric circuit, where in the undirected case, each hyperedge $e \in E$ acts as an imaginary unit such that the current flows from the vertex in e with the highest potential to that with the lowest and the resistance of e is $\boldsymbol{w}(e)$, and in the directed case, each hyperarc $e = (T_e, S_e) \in E$ acts as an imaginary unit such that the current flows from the vertex in T_e with the highest potential to the vertex in H_e with the lowest and the resistance of e is $\boldsymbol{w}(e)$. The effective resistance is equal to the potential difference of s and t when an unit electricity is injected to s and out from t. The effective resistance of hypergraphs have applications in network analysis and semi-supervised learning (see [12]).

An equivalent formulation of the effective resistance is the following.

LEMMA 6.2. *Let $G = (V, E, \boldsymbol{w})$ be an undirected/directed hypergraph. Then, we have*

$$R_G(s,t) = \min\{\boldsymbol{x}^\top L_G(\boldsymbol{x}) : \boldsymbol{x}(s) = 1, \boldsymbol{x}(t) = -1\}. \tag{6.5}$$

Proof. Introducing Lagrange multipliers λ_s and λ_t, an optimal solution $\boldsymbol{x} \in \mathbb{R}^V$ of (6.5) must satisfy $2L_G(\boldsymbol{x}) - \lambda_s \boldsymbol{e}_s - \lambda_t \boldsymbol{e}_t = 0$. Therefore, $L_G(\boldsymbol{x}) = \frac{1}{2}(\lambda_s \boldsymbol{e}_s + \lambda_t \boldsymbol{e}_t)$. This equality has a solution if $(\lambda_s \boldsymbol{e}_s + \lambda_t \boldsymbol{e}_t)^\top \mathbf{1} = 0$, which implies $\lambda_s + \lambda_t = 0$. Then, we can rewrite the condition as $L_G(\boldsymbol{x}) = \frac{\lambda}{2}(\boldsymbol{e}_s - \boldsymbol{e}_t)$ for some $\lambda \in \mathbb{R}$. Therefore, $\boldsymbol{x}$ is a potential corresponding to an st-flow. Furthermore, since $\boldsymbol{x}(s) = 1$ and $\boldsymbol{x}(t) = -1$, the flow is a unit flow and we must have $\lambda = 2$. Hence an optimal solution $\boldsymbol{x}$ satisfies $L_G(\boldsymbol{x}) = \boldsymbol{e}_s - \boldsymbol{e}_t$, which is in turn the optimality condition of (6.4). Evidently, the optimal value of (6.4) is equal to $\boldsymbol{x}^\top L_G(\boldsymbol{x})$.

An immediate corollary is the following.

COROLLARY 6.1. *If H is an ϵ-spectral sparsifier of G, then for distinct $s, t \in V$, we have $(1-\epsilon)R_H(s,t) \leq R_G(s,t) \leq (1+\epsilon)R_H(s,t)$.*

6.3 Faster Semi-Supervised Learning on Hypergraphs Zhang *et al.* [33] studied the following semi-supervised learning problem on directed hypergraphs. Suppose that we are given a directed hypergraph $G = (V, E, \boldsymbol{w})$, which represents a directional dependence of vertices. We are given labels $\boldsymbol{x}^*(u)$ of vertices u in $L \subseteq V$, and the task is to predict labels $\boldsymbol{x}(v) \in [0,1]$ in $v \in V \setminus L$, respecting the structure of the hypergraph. Intuitively, if a vertex s is downstream[3] of another vertex t in G, it is likely to hold that $\boldsymbol{x}(s) \leq \boldsymbol{x}(t)$. The formulation considered in [33] is as follows:

$$\begin{aligned} \min \quad & \sum_{e=(T_e,H_e)\in E} \boldsymbol{w}(e) \max_{u\in T_e} \max_{v \in H_e} \left([\boldsymbol{x}(u) - \boldsymbol{x}(v)]^+\right)^2 \\ \text{s.t.} \quad & \boldsymbol{x}(u) = \boldsymbol{x}^*(u) \quad (u \in L). \end{aligned}$$

Since the objective function is $\boldsymbol{x}^\top L_G(\boldsymbol{x})$, we can use our spectral sparsifiers for solving this problem faster. More specifically, in [33], they proposed a subgradient descent algorithm for this problem, which computes a subgradient at a given point in $O(\mathrm{size}(G))$ time. For simplicity, assume that G is r-regular. Then our hypergraph sparsification reduces the complexity of computing a subgradient to $\widetilde{O}(n^3 r)$ time, which is an improvement if $m = \omega(n^3)$.

References

[1] Z. Allen-Zhu, Z. Liao, and L. Orecchia. Spectral sparsification and regret minimization beyond matrix multiplicative updates. In *Proceedings of the 47th Annual ACM on Symposium on Theory of Computing (STOC)*, pages 237–245, 2015.

[2] J. Batson, D. A. Spielman, and N. Srivastava. Twice-Ramanujan sparsifiers. *SIAM Review*, 56(2):315–334, 2014.

[3] A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *Proceedings of the 28th Annual ACM symposium on Theory of computing (STOC)*, pages 47–55, 1996.

[4] A. A. Benczúr and D. R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *SIAM Journal on Computing*, 44(2):290–319, 2015.

[5] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[6] C. Chekuri and C. Xu. A note on approximate strengths of edges in a hypergraph. *arxiv*, 2017.

[7] M. Cheraghchi, A. Klivans, P. Kothari, and H. K. Lee. Submodular functions are noise stable. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1586–1592, 2012.

[8] M. B. Cohen, J. Kelner, J. Peebles, R. Peng, A. B. Rao, A. Sidford, and A. Vladu. Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC)*, pages 410–419, 2017.

[3] We say that s is downstream of t if there exists a directed path in G from t to s.

[9] M. K. de Carli Silva, N. J. A. Harvey, and C. M. Sato. Sparse sums of positive semidefinite matrices. *ACM Transactions on Algorithms*, 12(1):9:1–9:17, 2015.

[10] L. Devroye. *Non-Uniform Random Variate Generation*. Springer, 1986.

[11] V. Feldman, P. Kothari, and J. Vondrák. Representation, approximation and learning of submodular functions using low-rank decision trees. In *Proceedings of the 26th Annual Conference on Learning Theory (COLT)*, pages 711–740, 2013.

[12] K. Fujii, T. Soma, and Y. Yoshida. Polynomial-time algorithms for submodular Laplacian systems. *arxiv*, 2018.

[13] S. Fujishige and S. B. Patkar. Realization of set functions as cut functions of graphs and hypergraphs. *Discrete Mathematics*, 226(1):199 – 210, 2001.

[14] W. S. Fung, R. Hariharan, N. J. A. Harvey, and D. Panigrahi. A general framework for graph sparsification. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 71–80, 2011.

[15] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2-3):177–201, 1993.

[16] A. Gupta, M. Hardt, A. Roth, and J. Ullman. Privately releasing conjunctions and the statistical query barrier. *SIAM Journal on Computing*, 42(4):1494–1520, 2013.

[17] A. Jambulapati and A. Sidford. Efficient $\tilde{O}(n/\epsilon)$ spectral sketches for the Laplacian and its pseudoinverse. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2487–2503, 2018.

[18] D. R. Karger and M. S. Levine. Random sampling in residual graphs. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 63–66, 2002.

[19] D. Kogan and R. Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 6th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 367–376, 2015.

[20] I. Koutis, G. L. Miller, and R. Peng. Approaching optimality for solving SDD linear systems. *SIAM Journal on Computing*, 43(1):337–354, 2014.

[21] Y. T. Lee and H. Sun. Constructing linear-sized spectral sparsification in almost-linear time. In *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 250–269, 2015.

[22] Y. T. Lee and H. Sun. An SDP-based algorithm for linear-sized spectral sparsification. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC)*, pages 678–687, 2017.

[23] A. Louis. Hypergraph Markov operators, eigenvalues and approximation algorithms. In *Proceedings of the 47th Annual ACM on Symposium on Theory of Computing (STOC)*, pages 713–722, 2015.

[24] A. Madry. Fast approximation algorithms for cut-based problems in undirected graphs. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 245–254, 2010.

[25] Y. Mansour. An $O(n^{\log\log n})$ learning algorithm for DNF under the uniform distribution. *Journal of Computer and System Sciences*, 50(3):543–550, 1995.

[26] I. Newman and Y. Rabinovich. On multiplicative λ-approximations and some geometric applications. *SIAM Journal on Computing*, 42(3):855–883, 2013.

[27] S. Raskhodnikova and G. Yaroslavtsev. Learning pseudo-Boolean k-DNF and submodular functions. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1356–1368, 2013.

[28] J. Rodríguez. On the Laplacian eigenvalues and metric parameters of hypergraphs. *Linear and Multilinear Algebra*, 50(1):1–14, 2002.

[29] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[30] D. A. Spielman and S.-H. Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.

[31] D. A. Spielman and S.-H. Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014.

[32] Y. Yoshida. Cheeger inequalities for submodular transformations. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2019, to appear.

[33] C. Zhang, S. Hu, Z. G. Tang, and T.-H. H. Chan. Re-revisiting learning on hypergraphs: Confidence interval and subgradient method. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 4026–4034, 2017.

A A Hypergraph That Effective Resistance Fails to Sparsify

We provide an example for which a natural sparsification strategy based on effective resistance fails. More precisely, let us consider a sampling algorithm such that each hyperedge e is sampled with probability $p_e := \max_\pi R_{G_\pi}(e_\pi)$, where $R_{G_\pi}(e_\pi)$ denotes the effective resistance between the endpoints of e_π in the graph G_π (see Section 5 for the definitions of G_π and e_π). In the following, we show that $p_e \geq 1$ for any hyperedge e in the worst case, which means that the algorithm does not sparsify it at all.

Let r and n be positive integers. Let $V = \{s, t\} \cup U$, where U is a finite set of size n. Let $E = \{\{s, t\} \cup X : X \subseteq U, |X| = r\}$. Fix an arbitrary hyperedge $e =$

$\{s, t\} \cup X$. Define $\boldsymbol{x} \in \mathbb{R}^V$ as follows:

$$\boldsymbol{x}(v) := \begin{cases} 1 & v = s \\ 1 - \epsilon & v \in X \\ 1 + \epsilon & v \notin X \\ 0 & v = t, \end{cases}$$

where $\epsilon \in (0, 1)$ is a constant. Let π be an ordering of V in the descending order of $\boldsymbol{x}$ (break ties arbitrary). Then, G_π has only one edge connecting s and t, which is realized by e, and the other edges do not connect s and t in G_π. Therefore, $R_{G_\pi}(e_\pi) = 1$ for this ordering π. Since e was taken to be arbitrary, we must have $\max_\pi R_{G_\pi}(e_\pi) \geq 1$ for any hyperedge e.