Spectral Sparsification of Graphs*

Daniel A. Spielman
Department of Computer Science
Program in Applied Mathematics
Yale University

Shang-Hua Teng
Department of Computer Science
Viterbi School of Engineering
University of Southern California

July 22, 2010

Abstract

We introduce a new notion of graph sparsification based on spectral similarity of graph Laplacians: spectral sparsification requires that the Laplacian quadratic form of the sparsifier approximate that of the original. This is equivalent to saying that the Laplacian of the sparsifier is a good preconditioner for the Laplacian of the original.

We prove that every graph has a spectral sparsifier of nearly-linear size. Moreover, we present an algorithm that produces spectral sparsifiers in time $O\left(m \log^c m\right)$, where m is the number of edges in the original graph and c is some absolute constant. This construction is a key component of a nearly-linear time algorithm for solving linear equations in diagonally-dominant matrices.

Our sparsification algorithm makes use of a nearly-linear time algorithm for graph partitioning that satisfies a strong guarantee: if the partition it outputs is very unbalanced, then the larger part is contained in a subgraph of high conductance.

*This paper is the second in a sequence of three papers expanding on material that appeared first under the title "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems" [ST04]. The first paper, "A Local Clustering Algorithm for Massive Graphs and its Application to Nearly-Linear Time Graph Partitioning" [ST08a] contains graph partitioning algorithms that are used to construct the sparsifiers in this paper. The third paper, "Nearly-Linear Time Algorithms for Preconditioning and Solving Symmetric, Diagonally Dominant Linear Systems" [ST08b] contains the results on solving linear equations and approximating eigenvalues and eigenvectors.

This material is based upon work supported by the National Science Foundation under Grant Nos. 0325630, 0324914, 0634957, 0635102 and 0707522. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Shang-Hua Teng wrote part of this paper while at MSR-NE lab and Boston University.

1 Introduction

Graph sparsification is the task of approximating a graph by a sparse graph, and is often useful in the design of efficient approximation algorithms. Several notions of graph sparsification have been proposed. For example, Chew [Che89] was motivated by proximity problems in computational geometry to introduce graph spanners. Spanners are defined in terms of the *distance similarity* of two graphs: A spanner is a sparse graph in which the shortest-path distance between every pair of vertices is approximately the same in the original graph as in the spanner. Motivated by cut problems, Benczur and Karger [BK96] introduced a notion of sparsification that requires that for every set of vertices, the weight of the edges leaving that set should be approximately the same in the original graph as in the sparsifier.

Motivated by problems in numerical linear algebra and spectral graph theory, we introduce a new notion of sparsification that we call *spectral sparsification.* A spectral sparsifier is a subgraph of the original whose Laplacian quadratic form is approximately the same as that of the original graph on all real vector inputs. The Laplacian matrix[1] of a weighted graph $G = (V, E, w)$, where $w_{(u,v)}$ is the weight of edge (u, v), is defined by

$$L_G(u,v) = \begin{cases} -w_{(u,v)} & \text{if } u \neq v \\ \sum_z w_{(u,z)} & \text{if } u = v. \end{cases}$$

It is better understood by its quadratic form, which on $x \in \mathbb{R}^V$ takes the value

$$x^T L_G x = \sum_{(u,v)\in E} w_{(u,v)} \left(x(u) - x(v)\right)^2. \tag{1}$$

We say that $\widetilde{G}$ is a *σ-spectral approximation* of G if for all $x \in \mathbb{R}^V$

$$\frac{1}{\sigma} x^T L_{\widetilde{G}} x \leq x^T L_G x \leq \sigma x^T L_{\widetilde{G}} x. \tag{2}$$

Our notion of sparsification captures the *spectral similarity* between a graph and its sparsifiers. It is a stronger notion than the cut sparsification of Benczur and Karger: the cut-sparsifiers constructed by Benczur and Karger [BK96] are only required to satisfy these inequalities for all $x \in \{0,1\}^V$. In Section 5 we present an example demonstrating that these notions of approximation are in fact different.

Our main result is that every weighted graph has a spectral sparsifier with $\widetilde{\mathcal{O}}(n)$ edges that can be computed in $\widetilde{\mathcal{O}}(m)$ time, where we recall that $\widetilde{\mathcal{O}}(f(n))$ means $O(f(n) \log^c f(n))$, for some constant c. In particular, we prove that for every weighted graph $G = (V, E, w)$ and every $\epsilon > 0$, there is a re-weighted subgraph of G with $\widetilde{\mathcal{O}}\left(n/\epsilon^2\right)$ edges that is a $(1+\epsilon)$ approximation of G. Moreover, we show how to find such a subgraph in $\widetilde{\mathcal{O}}(m)$ time, where $n = |V|$ and $m = |E|$. The constants and powers of logarithms hidden in the $\widetilde{\mathcal{O}}$-notation in the statement of our results are quite large. Our goal in this paper is not to produce sparsifiers with optimal parameters, but rather just to prove that spectral sparsifiers with a nearly-linear number of edges exist and that they can be found in nearly-linear time.

[1] For more information on the Laplacian matrix of a graph, we refer the reader to one of [Bol98, Moh91, GR01, Chu97].

Our sparsification algorithm makes use of a nearly-linear time graph partitioning algorithm, `ApproxCut`, that we develop in Section 8 and which may be of independent interest. On input a target conductance ϕ, `ApproxCut` always outputs a set of vertices of conductance less than ϕ. With high probability, if the set it outputs is small then its complement is contained in a subgraph of conductance at least $\Omega(\phi^2/\log^4 m)$.

2 The Bigger Picture

This paper arose in our efforts to design nearly-linear time algorithms for solving diagonally-dominant linear systems, and is the second in a sequence of three papers on the topic. In the first paper [ST08a], we develop fast routines for partitioning graphs, which we then use in our algorithms for building sparsifiers. In the last paper [ST08b], we show how to use sparsifiers to build preconditioners for diagonally-dominant matrices and thereby solve linear equations in such matrices in nearly-linear time. Koutis, Miller and Peng [KMP10] have recently developed an algorithm for solving such systems of linear equations in time $O(m \log^2 n)$ that does not rely upon the sparsifiers of the present paper.

The quality of a preconditioner is measured by the relative condition number, which for the Laplacian matrices of a graph G and its sparsifier $\widetilde{G}$ is

$$\kappa(G, \widetilde{G}) \stackrel{\text{def}}{=} \left(\max_x \frac{x^T L_G x}{x^T L_{\widetilde{G}} x}\right) \Big/ \left(\min_x \frac{x^T L_G x}{x^T L_{\widetilde{G}} x}\right)$$

So, if $\widetilde{G}$ is a σ-spectral approximation of G then $\kappa(G, \widetilde{G}) \leq \sigma^2$. This means that an iterative solver such as the Preconditioned Conjugate Gradient [Axe85] can solve a linear system in the Laplacian of G to accuracy ϵ by solving $O(\sigma \log(1/\epsilon))$ linear systems in $\widetilde{G}$ and performing as many multiplications by G. As a linear system in a matrix with m non-zero entries may be solved in time $O(nm)$ by using the Conjugate Gradient as a direct method [TB97, Theorem 28.3], the use of the sparsifiers in this paper alone provides an algorithm for solving linear systems in L_G to ϵ-accuracy in time $\widetilde{\mathcal{O}}\left(n^2 \log(1/\epsilon)\right)$, which is nearly optimal when the Laplacian matrix has $\Omega(n^2)$ non-zero entries. In our paper on solving linear equations [ST08b], we show how to get the time bound down to $\widetilde{\mathcal{O}}\left(m \log(1/\epsilon)\right)$, where m is the number of non-zero entries in L_G.

3 Outline

In Section 4, we present technical background required for this paper, and maybe even for the rest of this outline. In Section 5, we present three examples of graphs and their sparsifiers. These examples help motivate key elements of our construction.

There are three components to our algorithm for sparsifying graphs. The first is a random sampling procedure. In Section 6, we prove that this procedure produces good spectral sparsifiers for graphs of high conductance. So that we may reduce the problem of sparsifying arbitrary graphs to that of sparsifying graphs of high conductance, we require a fast algorithm for partitioning a graph into parts of high conductance without removing too many edges. In Section 7, we first prove that such partitions exist, and use them to prove the existence of spectral sparsifiers for all unweighted graphs. In Section 8, we then build on tools from [ST08a] to develop a graph partitioning procedure that suffices. We use this procedure in Section 9 to construct a nearly-linear time algorithm for sparsifying unweighted graphs. We show how to use this algorithm to sparsify weighted graphs in Section 10.

We conclude in Section 11 by surveying recent improvements that have been made in both sparsification and in the partitioning routines on which the present paper depends.

4 Background and Notation

By log we always mean the logarithm base 2, and we denote the natural logarithm by ln.

As we spend this paper studying spectral approximations, we will say "σ-approximation" instead of "σ-spectral approximation" wherever it won't create confusion.

We may express (2) more compactly by employing the notation $A \preccurlyeq B$ to mean

$$x^T A x \leq x^T B x, \quad \text{for all } x \in \mathbb{R}^V.$$

Inequality (2) is then equivalent to

$$\frac{1}{\sigma} L_{\widetilde{G}} \preccurlyeq L_G \preccurlyeq \sigma L_{\widetilde{G}}. \tag{3}$$

We will overload notation by writing $G \preccurlyeq \widetilde{G}$ for graphs G and $\widetilde{G}$ to mean $L_G \preccurlyeq L_{\widetilde{G}}$.

For two graphs G and H, we write

$$G + H$$

to indicate the graph whose Laplacian is $L_G + L_H$. That is, the weight of every edge in $G + H$ is the sum of the weights of the corresponding edges in G and H. We will use this notation even if G and H have different vertex sets. For example, if their vertex sets are disjoint, then their sum is simply the disjoint union of the graphs. It is immediate that $G \preccurlyeq \widetilde{G}$ and $H \preccurlyeq \widetilde{H}$ imply

$$G + H \preccurlyeq \widetilde{G} + \widetilde{H}.$$

In many portions of this paper, we will consider vertex-induced subgraphs of graphs. *When we take subgraphs, we always preserve the identity of vertices.* This enables us to sum inequalities on the different subgraphs to say something about the original.

For an unweighted graph $G = (V, E)$, we will let d_v denote the degree of vertex v. For S and T disjoint subsets of V, we let $E(S, T)$ denote the set of edges in E connecting one vertex of S with one vertex of T. We let $G(S)$ denote the subgraph of G induced on the vertices in S: the graph with vertex set S containing the edges of E between vertices in S.

For $S \subseteq V$, we define $\mathrm{Vol}\,(S) = \sum_{i \in S} d_i$. Observe that $\mathrm{Vol}\,(V) = 2m$ if G has m edges. The conductance of a set of vertices S, written $\Phi_G\,(S)$, is often defined by

$$\Phi_G\,(S) \stackrel{\text{def}}{=} \frac{|E(S, V - S)|}{\min\left(\mathrm{Vol}\,(S), \mathrm{Vol}\,(V - S)\right)}.$$

The conductance of G is then given by

$$\Phi_G \stackrel{\text{def}}{=} \min_{\emptyset \neq S \subset V} \Phi\,(S)\,.$$

The conductance of a graph is related to the smallest non-zero eigenvalue of its Laplacian matrix, but is even more strongly related to the smallest non-zero eigenvalue of its Normalized

Laplacian matrix (see [Chu97]), whose definition we now recall. Let D be the diagonal matrix whose v-th diagonal is d_v. The Normalized Laplacian of the graph G, written $\mathcal{L}_G$, is defined by

$$\mathcal{L}_G = D^{-1/2} L_G D^{-1/2}.$$

It is well-known that both L_G and $\mathcal{L}_G$ are positive semi-definite matrices, with smallest eigenvalue zero. The eigenvalue zero has multiplicity one if an only if the graph G is connected, in which case the eigenvector of L_G with eigenvalue zero is the constant vector (see [Bol98, page 269], or derive from (1)).

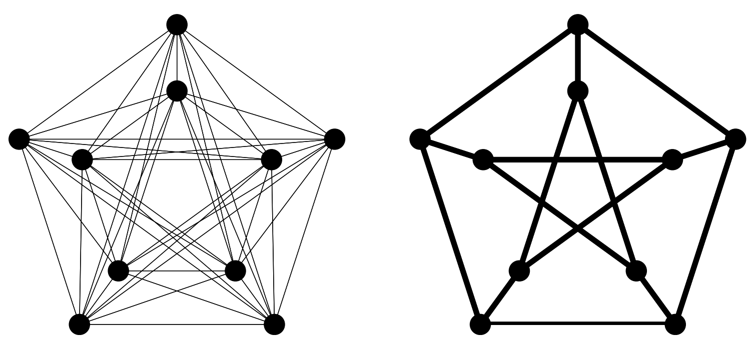
Our analysis exploits a discreet version of Cheeger's inequality[Che70] (see [Chu97, SJ89, DS91]), which relates the smallest non-zero eigenvalue of $\mathcal{L}_G$, written $\lambda_2(\mathcal{L}_G)$, to the conductance of G.

Theorem 4.1 (Cheeger's Inequality)**.**

$$2\Phi_G \geq \lambda_2(\mathcal{L}_G) \geq \Phi_G^2/2.$$

5 A few examples

5.1 Example 1: Complete Graph



G: The complete graph on 10 vertices

$\widetilde{G}$: A $\sqrt{5/2}$-approximation of G

We first consider what a sparsifier of the complete graph should look like. Let G be the complete graph on n vertices. All non-zero eigenvalues of L_G equal n. So, for every unit vector x orthogonal to the all-1s vector,

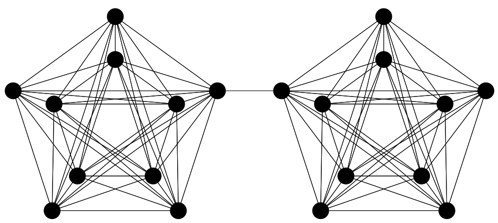
$$x^T L_G x = n.$$

From Cheeger's inequality, one may prove that graphs with constant conductance, called expanders, have a similar property. Spectrally speaking, the best of them are the Ramanujan graphs [LPS88, Mar88], which are d-regular graphs all of whose non-zero Laplacian eigenvalues lie between $d - 2\sqrt{d-1}$ and $d + 2\sqrt{d+1}$. So, if we let $\widetilde{G}$ be a Ramanujan graph in which every edge has been given weight n/d, then for every unit vector x orthogonal to the all-1s vector,

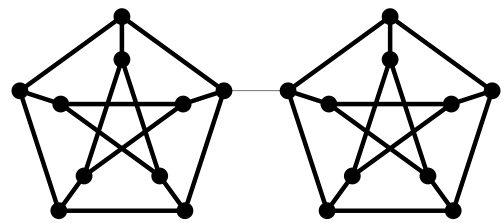
$$x^T L_{\widetilde{G}} x \in \left[n - \frac{2n\sqrt{d-1}}{d}, n + \frac{2n\sqrt{d-1}}{d} \right].$$

Thus, $\widetilde{G}$ is a $\left(1 - 2\sqrt{d-1}/d\right)^{-1}$-approximation of G.

5.2 Example 2: Joined Complete Graphs



G: Two complete graphs joined by an edge.



$\widetilde{G}$: A good approximation of G. Thicker edges indicate edges of weight 3

Next, consider a graph on $2n$ vertices obtained by joining two complete graphs on n vertices by a single edge, e. Let V_1 and V_2 be the vertex sets of the two complete graphs. We claim that a good sparsifier for G may be obtained by setting $\widetilde{G}$ to be the edge e with weight 1, plus (n/d) times a Ramanujan graph on each vertex set. To prove this, let G_1 and G_2 denote the complete graphs on V_1 and V_2, and let G_3 denote the graph just consisting of the edge e. Similarly, let $\widetilde{G}_1$ and $\widetilde{G}_2$ denote (n/d) times a Ramanujan graph on each vertex set, and let $\widetilde{G}_3 = G_3$. Recalling the addition we defined on graphs, we have

$$\begin{aligned} G &= G_1 + G_2 + G_3, \quad \text{and} \\ \widetilde{G} &= \widetilde{G}_1 + \widetilde{G}_2 + \widetilde{G}_3. \end{aligned}$$

We already know that for $\sigma = \left(1 - 2\sqrt{d-1}/d\right)^{-1}$, and $i \in \{1, 2\}$

$$\frac{1}{\sigma}\widetilde{G}_i \preccurlyeq G_i \preccurlyeq \sigma \widetilde{G}_i.$$

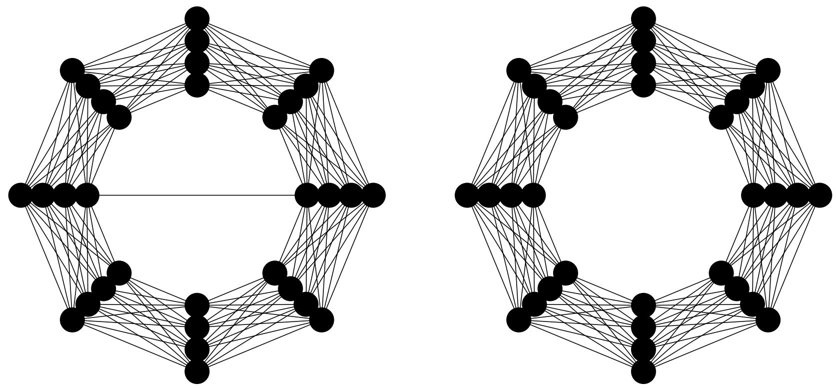
As $\widetilde{G}_3 = G_3$, we have

$$G = G_1 + G_2 + G_3 \preccurlyeq \sigma\widetilde{G}_1 + \sigma\widetilde{G}_2 + \widetilde{G}_3 \preccurlyeq \sigma\widetilde{G}_1 + \sigma\widetilde{G}_2 + \sigma\widetilde{G}_3 = \sigma\widetilde{G}.$$

The other inequality follows by similar reasoning. This example demonstrates both the utility of using edges with different weights, even when sparsifying unweighted graphs, and how we can combine sparsifiers of subgraphs to sparsify an entire graph. Also observe that every sparsifier of G must contain the edge e, while no other edge is particularly important.

5.3 Example 3: Distinguishing cut sparsifiers from spectral sparsifiers

Our last example will demonstrate the difference between our notion of sparsification and that of Benczur and Karger. We will describe graphs G and $\widetilde{G}$ for which $\widetilde{G}$ is not a σ-approximation of G for any small σ, but it is a very good sparsifier of G under the definition considered by Benczur and Karger. The vertex set V will be $\{0, \ldots, n-1\} \times \{1, \ldots, k\}$, where n is even. The graph $\widetilde{G}$ will consist of n complete bipartite graphs, connecting all pairs of vertices (u, i) and (v, j) where $v = u \pm 1 \mod n$. The graph G will be identical to the graph $\widetilde{G}$, except that it will have one additional edge e from vertex $(0, 1)$ to vertex $(n/2, 1)$. As the minimum cut of G has size $2k$, and $\widetilde{G}$ only differs by one edge, $\widetilde{G}$ is a $(1 + 1/2k)$-approximation of G in the



G: $n = 8$ sets of $k = 4$ vertices arranged in a ring and connected by complete bipartite graphs, plus one edge across.

$\widetilde{G}$: A good cut sparsifier of G, but a poor spectral sparsifier

notion considered by Benczur and Karger. To show that $\widetilde{G}$ is a poor spectral approximation of G, consider the vector x given by

$$x(u, i) = \min(u, n - u).$$

One can verify that

$$x^T L_{\widetilde{G}} x = nk^2, \quad \text{while} \quad x^T L_G x = nk^2 + (n/2)^2.$$

So, inequality (2) is not satisfied for any σ less than $1 + n/4k^2$.

6 Sampling Graphs

In this section, we show that if a graph has high conductance, then it may be sparsified by a simple random sampling procedure. The sampling procedure involves assigning a probability $p_{i,j}$ to each edge (i, j), and then selecting edge (i, j) to be in the graph $\widetilde{G}$ with probability $p_{i,j}$. When edge (i, j) is chosen to be in the graph, we multiply its weight by $1/p_{i,j}$. As the graph is undirected, we implicitly assume that $p_{i,j} = p_{j,i}$. Let A denote the adjacency matrix of the original graph G, and $\widetilde{A}$ the adjacency matrix of the sampled graph $\widetilde{G}$. This procedure guarantees that

$$\mathbf{E}\left[\widetilde{A}\right] = A.$$

Sampling procedures of this form were examined by Benczur and Karger [BK96] and Achlioptas and McSherry [AM01]. Achlioptas and McSherry analyze the approximation obtained by such a procedure through a bound on the norm of a random matrix of Füredi and Komlós [FK81]. As their bound does not suffice for our purposes, we tighten it by refining the analysis of Füredi and Komlós.

If $\widetilde{G}$ is going to be a sparsifier for G, then we must be sure that every vertex in $\widetilde{G}$ has edges attached to it. We guarantee this by requiring that, for some parameter $\Upsilon > 1$,

$$p_{i,j} = \min\left(1, \frac{\Upsilon}{\min(d_i, d_j)}\right), \quad \text{for all edges } (i, j). \tag{4}$$

The parameter Υ controls the number of edges we expect to find in the graph, and will be set to at least $\Omega(\log n)$ to ensure that every vertex has an attached edge.

We will show that if G has high conductance and (4) is satisfied for a sufficiently large Υ, then $\widetilde{G}$ will be a good sparsifier of G with high probability. The actual theorem that we prove is slightly more complicated, as it considers the case where we only apply the sampling on a subgraph of G.

Theorem 6.1 (Sampling High-Conductance Graphs)**.** *Let $\epsilon, p \in (0, 1/2)$ and let $G = (V, E)$ be an unweighted graph whose smallest non-zero normalized Laplacian eigenvalue is at least λ. Let S be a subset of the vertices of G, let F be the edges in $G(S)$, and let $H = E - F$ be the rest of the edges. Let*

$$(S, \widetilde{F}) = \texttt{Sample}((S, F), \epsilon, p, \lambda),$$

and let $\widetilde{G} = (V, \widetilde{F} \cup H)$. Then, with probability at least $1 - p$,

(S.1) $\widetilde{G}$ is a $(1+\epsilon)$-approximation of G, and

(S.2) The number of edges in $\widetilde{F}$ is at most

$$\frac{288 \max\left(\log_2(3/p), \log_2 n\right)^2}{(\epsilon\lambda)^2} |S|.$$

> $\widetilde{G} = \texttt{Sample}(G, \epsilon, p, \lambda)$
>
> 1. Set $k = \max\left(\log_2(3/p), \log_2 n\right)$.
> 2. Set $\Upsilon = \left(\frac{12k}{\epsilon\lambda}\right)^2$.
> 3. For every edge (i, j) in G, set $p_{i,j} = \min\left(1, \frac{\Upsilon}{\min(d_i, d_j)}\right)$.
> 4. For every edge (i, j) in G, with probability $p_{i,j}$ put an edge of weight $1/p_{i,j}$ between vertices (i, j) into $\widetilde{G}$.

Let D be the diagonal matrix of degrees of vertices of G. To prove Theorem 6.1, we establish that the 2-norm of $D^{-1/2}(L_G - L_{\widetilde{G}})D^{-1/2}$ is probably small[2], and then apply the following lemma.

Lemma 6.2. *Let L be the Laplacian matrix of a connected graph G, $\widetilde{L}$ be the Laplacian of $\widetilde{G}$, and let D be the diagonal matrix of degrees of G. If*

1. $\lambda_2\left(D^{-1/2} L D^{-1/2}\right) \geq \lambda$, *and*
2. $\left\| D^{-1/2}(L - \widetilde{L}) D^{-1/2} \right\| \leq \epsilon$,

then $\widetilde{G}$ is a σ-approximation of G for

$$\sigma = \frac{\lambda}{\lambda - \epsilon}.$$

[2]Recall that the 2-norm of a symmetric matrix is the largest absolute value of its eigenvalues.

Proof. Let x be any vector and let $y = D^{1/2}x$. By assumption, G is connected and so the nullspace of the normalized Laplacian $D^{-1/2}LD^{-1/2}$ is spanned by $D^{1/2}\mathbf{1}$. Let z be the projection of y orthogonal to $D^{1/2}\mathbf{1}$, so

$$x^T Lx = y^T D^{-1/2}LD^{-1/2}y = z^T \left(D^{-1/2}LD^{-1/2}\right) z \geq \lambda \left\|z\right\|^2. \tag{5}$$

We compute

$$\begin{aligned}
x^T \widetilde{L}x &= y^T D^{-1/2}\widetilde{L}D^{-1/2}y \\
&= z^T D^{-1/2}\widetilde{L}D^{-1/2}z \\
&= z^T D^{-1/2}LD^{-1/2}z + z^T D^{-1/2}(\widetilde{L} - L)D^{-1/2}z \\
&= z^T D^{-1/2}LD^{-1/2}z \left(1 + \frac{z^T D^{-1/2}(\widetilde{L} - L)D^{-1/2}z}{z^T D^{-1/2}LD^{-1/2}z}\right) \\
&\geq z^T D^{-1/2}LD^{-1/2}z \left(1 - \frac{\epsilon \left\|z\right\|^2}{\lambda \left\|z\right\|^2}\right) \quad \text{(by assumption 2 and (5))} \\
&= \left(\frac{\lambda - \epsilon}{\lambda}\right) x^T Lx. \quad \text{(again by (5))}
\end{aligned}$$

We may similarly show that

$$x^T \widetilde{L}x \leq \left(\frac{\lambda + \epsilon}{\lambda}\right) x^T Lx \leq \left(\frac{\lambda}{\lambda - \epsilon}\right) x^T Lx.$$

The lemma follows from these inequalities. □

Let A be the adjacency matrix of G and let $\widetilde{A}$ be the adjacency matrix of $\widetilde{G}$. For each edge (i, j),

$$\widetilde{A}_{i,j} = \begin{cases} 1/p_{i,j} & \text{with probability } p_{i,j} \text{ and} \\ 0 & \text{with probability } 1 - p_{i,j}. \end{cases}$$

To prove Theorem 6.1, we will observe that

$$\left\|D^{-1/2}(L - \widetilde{L})D^{-1/2}\right\| \leq \left\|D^{-1/2}(A - \widetilde{A})D^{-1/2}\right\| + \left\|D^{-1/2}(D - \widetilde{D})D^{-1/2}\right\|,$$

where $\widetilde{D}$ is the diagonal matrix of the diagonal entries of $\widetilde{L}$. It will be easy to bound the second of these terms, so we defer that part of the proof to the end of the section. A bound on the first term comes from the following lemma.

Lemma 6.3 (Random Subgraph)**.** *For all even integers k,*

$$\Pr\left[\left\|D^{-1/2}(\widetilde{A} - A)D^{-1/2}\right\| \geq \frac{2kn^{1/k}}{\sqrt{\Upsilon}}\right] \leq 2^{-k}.$$

Our proof of this lemma applies a modification of techniques introduced by Füredi and Komlós [FK81] (See also the paper by Vu [Vu07] that corrects some bugs in their work). However, they consider the eigenvalues of random graphs in which every edge can appear. Some interesting modifications are required to make an argument such as ours work when downsampling a graph that may already be sparse. We remark that without too much work one can generalize Theorem 6.1 so that it applies to weighted graphs.

Proof of Lemma 6.3. To simplify notation, define

$$\Delta = D^{-1}(\widetilde{A} - A),$$

so for each edge (i, j),

$$\Delta_{i,j} = \begin{cases} \frac{1}{d_i}(\frac{1}{p_{i,j}} - 1) & \text{with probability } p_{i,j}, \text{ and} \\ -\frac{1}{d_i} & \text{with probability } 1 - p_{i,j}. \end{cases}$$

Note that $D^{-1/2}(\widetilde{A} - A)D^{-1/2}$ has the same eigenvalues as Δ. So, it suffices to bound the absolute values of the eigenvalues of Δ. Rather than trying to upper bound the eigenvalues of Δ directly, we will upper bound a power of Δ's trace. As the trace of a matrix is the sum of its eigenvalues, $\mathrm{Tr}\left(\Delta^k\right)$ is an upper bound on the kth power of every eigenvalue of Δ, for every even power k.

Lemma 6.4 implies that, for even k,

$$\frac{nk^k}{\Upsilon^{k/2}} \geq \mathbf{E}\left[\mathrm{Tr}\left(\Delta^k\right)\right] \geq \mathbf{E}\left[\lambda_{max}\left(\Delta^k\right)\right].$$

Applying Markov's inequality, we obtain

$$\Pr\left[\mathrm{Tr}\left(\Delta^k\right) \geq 2^k \frac{nk^k}{\Upsilon^{k/2}}\right] \leq 1/2^k.$$

Recalling that the eigenvalues of Δ^k are the k-th powers of the eigenvalues of Δ, and taking k-th roots, we conclude

$$\Pr\left[\left\|D^{-1/2}(\widetilde{A} - A)D^{-1/2}\right\| \geq 2\frac{n^{1/k}k}{\Upsilon^{1/2}}\right] \leq 1/2^k.$$

□

Lemma 6.4. *For even k,*

$$\mathbf{E}\left[\mathrm{Tr}\left(\Delta^k\right)\right] \leq \frac{nk^k}{\Upsilon^{k/2}}.$$

Proof. Recall that the (v_0, v_k) entry of Δ^k satisfies

$$\left(\Delta^k\right)_{v_0,v_k} = \sum_{v_1,\ldots,v_{k-1}} \prod_{i=1}^{k} \Delta_{v_{i-1},v_i}.$$

Taking expectations, we obtain

$$\mathbf{E}\left[\left(\Delta^k\right)_{v_0,v_k}\right] = \sum_{v_1,\ldots,v_{k-1}} \mathbf{E}\left[\prod_{i=1}^{k} \Delta_{v_{i-1},v_i}\right]. \tag{6}$$

We will now describe a way of coding every sequence $v_1, \ldots, v_{k-1}$ that could possibly contribute to the sum. Of course, any sequence containing a consecutive pair (v_{i-1}, v_i) for which Δ_{v_{i-1},v_i} is always zero will contribute zero to the sum. So, for a sequence to have a non-zero contribution, each consecutive pair (v_{i-1}, v_i) must be an edge in the graph A. Thus, we can identify every sequence with non-zero contribution with a walk on the graph A from vertex v_0 to vertex v_k.

The first idea in our analysis is to observe that most of the terms in this sum are zero. The reason is that, for all v_i and v_j

$$\mathbf{E}\left[\Delta_{v_i,v_j}\right] = 0.$$

As Δ_{v_i,v_j} is independent of every term in Δ other than Δ_{v_j,v_i}, we see that the term

$$\mathbf{E}\left[\prod_{i=1}^{k} \Delta_{v_{i-1},v_i}\right], \tag{7}$$

corresponding to $v_1, \ldots, v_{k-1}$, will be zero unless each edge (v_{i-1}, v_i) appears at least twice (in either direction).

We now describe a method for coding all walks in which each edges appears at least twice. We set T to be the set of time steps i at which the edge between v_{i-1} and v_i does not appear earlier in the walk (in either direction). Note that 1 is always an element of T. We then let τ denote the map from $[k] - T \to T$, indicating for each time step not in T the time step in which the edge traversed first appeared (regardless of in which direction it is traversed). Note that we need only consider the cases in which $|T| \leq k/2$, as otherwise some edge appears only once in the walk. To finish our description of a walk, we need a map

$$\sigma : T \to \{1, \ldots, n\},$$

indicating the vertex encountered at each time $i \in T$.

For example, for the walk

Step	0	1	2	3	4	5	6	7	8	9	10
Vertex	a	b	c	d	b	c	d	b	e	b	a

,

we get

$$T = \{1,2,3,4,8\} \qquad \tau : \begin{array}{r} 5 \mapsto 2 \\ 6 \mapsto 3 \\ 7 \mapsto 4 \\ 9 \mapsto 8 \\ 10 \mapsto 1 \end{array} \qquad \sigma : \begin{array}{r} 1 \mapsto b \\ 2 \mapsto c \\ 3 \mapsto d \\ 4 \mapsto b \\ 8 \mapsto e \end{array}$$

Using T, τ and σ, we can inductively reconstruct the sequence $v_1, \ldots, v_{k-1}$ by the rules

- if $i \in T$, $v_i = \sigma(i)$,

- if $i \notin T$, and $v_{i-1} = v_{\tau(i)-1}$, then $v_i = v_{\tau(i)}$, and
- if $i \notin T$, and $v_{i-1} = v_{\tau(i)}$, then $v_i = v_{\tau(i)-1}$.

If $v_{i-1} \notin \left\{v_{\tau(i)}, v_{\tau(i)-1}\right\}$, then the tuple (T, τ, σ) does not properly code a walk on the graph of A. We will call σ a *valid assignment* for T and τ if the above rules do produce a walk on the graph of A from v_0 to v_k.

We have

$$\mathbf{E}\left[\left(\Delta^k\right)_{v_0,v_k}\right] = \sum_{T,\tau} \sum_{\sigma \text{ valid for } T \text{ and } \tau} \mathbf{E}\left[\prod_{i=1}^{k} \Delta_{v_{i-1},v_i}\right],$$

(where $(v_1, \ldots, v_{k-1})$ is the sequence encoded by (T, τ, σ))

$$= \sum_{T,\tau} \sum_{\sigma \text{ valid for } T \text{ and } \tau} \prod_{s \in T} \mathbf{E}\left[\Delta_{v_{s-1},v_s} \prod_{i:\tau(i)=s} \Delta_{v_{i-1},v_i}\right]. \tag{8}$$

Each of the terms

$$\mathbf{E}\left[\Delta_{v_{s-1},v_s} \prod_{i:\tau(i)=s} \Delta_{v_{i-1},v_i}\right]$$

is independent of the others, and involves a product of the terms Δ_{v_{s-1},v_s} and $\Delta_{v_s,v_{s-1}}$. In Lemma 6.6, we will prove that

$$\mathbf{E}\left[\Delta_{v_{s-1},v_s} \prod_{i:\tau(i)=s} \Delta_{v_{i-1},v_i}\right] \leq \frac{1}{\Upsilon^{|\{i:\tau(i)=s\}|}} \frac{1}{d_{v_{s-1}}}, \tag{9}$$

which implies

$$\sum_{\sigma \text{ valid for } T \text{ and } \tau} \prod_{s \in T} \mathbf{E}\left[\Delta_{v_{s-1},v_s} \prod_{i:\tau(i)=s} \Delta_{v_{i-1},v_i}\right] \leq \frac{1}{\Upsilon^{k-|T|}} \sum_{\sigma \text{ valid for } T \text{ and } \tau} \prod_{s \in T} \frac{1}{d_{v_{s-1}}}. \tag{10}$$

To bound the sum of products on the right hand-side of (10), fix T and τ and consider the following random process for generating a valid σ and corresponding walk: go through the elements of T in order. For each $s \in T$, pick $\sigma(s)$ to be a random neighbor of the $s-1$st vertex in the walk. If possible, continue the walk according to τ until it reaches the next step in T. If the process produces a valid σ, return it. Otherwise, return nothing. The probability that any particular valid σ will be returned by this process is

$$\prod_{s \in T} \frac{1}{d_{v_{s-1}}}.$$

So,

$$\sum_{\sigma \text{ valid for } T \text{ and } \tau} \prod_{s \in T} \frac{1}{d_{v_{s-1}}} \leq 1. \tag{11}$$

As there are at most at most 2^k choices for T, and at most $|T|^{k-|T|} \leq |T|^k$ choices for τ, we may combine inequalities (10) and (11) with (8) to obtain

$$\mathbf{E}\left[\left(\Delta^k\right)_{v_0,v_k}\right] \leq \frac{(2\,|T|)^k}{\Upsilon^{k-|T|}} \leq \frac{k^k}{\Upsilon^{k/2}}. \quad (\text{using } |T| \leq k/2)$$

The lemma now follows from

$$\mathbf{E}\left[\mathrm{Tr}\left(\Delta^k\right)\right] = \sum_{v_0=1}^{n} \mathbf{E}\left[\left(\Delta^k\right)_{v_0,v_0}\right].$$

□

Claim 6.5.

$$|\Delta_{i,j}| \leq 1/\Upsilon.$$

Proof. If $p_{i,j} = 1$, then $\Delta_{i,j} = 0$. If not, then we have $\Upsilon/\min(d_i, d_j) = p_{i,j} < 1$. With probability $1 - p_{i,j}$,

$$|\Delta_{i,j}| = \frac{1}{d_i} \leq \frac{1}{\min(d_i, d_j)} \leq 1/\Upsilon.$$

On the other hand, with probability $p_{i,j}$,

$$\Delta_{i,j} = \frac{1}{d_i}\left(\frac{1}{p_{i,j}} - 1\right) \leq \frac{1}{d_i}\frac{1}{p_{i,j}} \leq \frac{1}{\min(d_i, d_j)}\frac{1}{p_{i,j}} = 1/\Upsilon.$$

As $\Delta_{i,j} \geq 0$ in this case, we have established $|\Delta_{i,j}| \leq 1/\Upsilon$. □

Lemma 6.6. *For all edges (r, t) and integers $k \geq 1$ and $l \geq 0$,*

$$\mathbf{E}\left[\Delta_{r,t}^k \Delta_{t,r}^l\right] \leq \frac{1}{\Upsilon^{k+l-1}}\frac{1}{d_r}.$$

Proof. First, if $p_{i,j} = 1$, then $\Delta_{i,j} = 0$. Second, if $k + l = 1$, $\mathbf{E}\left[\Delta_{r,t}^k \Delta_{t,r}^l\right] = 0$. So, we may restrict our attention to the case where $k + l \geq 2$ and $p_{i,j} < 1$, which by (4) implies $p_{i,j} = \Upsilon/\min(d_r, d_t)$. Claim 6.5 tells us that for $k \geq 1$,

$$\mathbf{E}\left[\Delta_{r,t}^k \Delta_{t,r}^l\right] \leq \frac{1}{\Upsilon}\mathbf{E}\left[\Delta_{r,t}^{k-1} \Delta_{t,r}^l\right].$$

A similar statement may be made for $l \geq 1$. So, it suffices to prove the lemma in the case $k + l = 2$.

As $\Delta_{r,t} = (\widetilde{A}_{r,t} - 1)/d_r$ and $\Delta_{t,r} = (\widetilde{A}_{r,t} - 1)/d_t$, we have

$$\begin{aligned}
\mathbf{E}\left[\Delta_{r,t}^k \Delta_{t,r}^l\right] &= \frac{1}{d_r^k d_t^l} \mathbf{E}\left[(\widetilde{A}_{r,t} - 1)^{k+l}\right] \\
&= \frac{1}{d_r^k d_t^l} \left(p_{r,t} \left(\frac{1 - p_{r,t}}{p_{r,t}}\right)^2 + (1 - p_{r,t})\right) && \text{(using } k + l = 2\text{)} \\
&= \frac{1}{d_r^k d_t^l} \left(\frac{1 - p_{r,t}}{p_{r,t}}\right) \\
&\leq \frac{1}{d_r^k d_t^l} \left(\frac{1}{p_{r,t}}\right) \\
&= \frac{1}{d_r^k d_t^l} \left(\frac{\min(d_r, d_t)}{\Upsilon}\right).
\end{aligned}$$

In the case $k = 1$, $l = 1$, we finish the proof by

$$\frac{\min(d_r, d_t)}{d_r d_t} = \frac{1}{\max(d_r, d_t)} \leq \frac{1}{d_r},$$

and in the case $k = 2$, $l = 0$ by

$$\frac{\min(d_r, d_t)}{d_r^2} \leq \frac{1}{d_r}.$$

□

This finishes the proofs of Lemmas 6.4 and 6.3. We now turn to the last ingredient we will need for the proof of Theorem 6.1, a bound on the norm of the difference of the degree matrices.

Lemma 6.7. *Let G be a graph and let $\widetilde{G}$ be obtained by sampling G with probabilities $p_{i,j}$ that satisfy* (4)*. Let D be the diagonal matrix of degrees of G, and let $\widetilde{D}$ be the diagonal matrix of weighed degrees of $\widetilde{G}$. Then,*

$$\Pr\left[\left\| D^{-1/2}(D - \widetilde{D})D^{-1/2} \right\| \geq \epsilon\right] \leq 2ne^{-\Upsilon\epsilon^2/3}.$$

Proof. Let $\tilde{d}_i$ be the weighted degree of vertex i in $\widetilde{G}$. As D and $\widetilde{D}$ are diagonal matrices,

$$\left\| D^{-1/2}(D - \widetilde{D})D^{-1/2} \right\| = \max_i \left| 1 - \frac{\tilde{d}_i}{d_i} \right|.$$

As the expectation of $\tilde{d}_i$ is d_i and $\tilde{d}_i$ is a sum of d_i random variables each of which is always 0 or some value less than d_i/Υ, we may apply the variant of the Chernoff bound given in Theorem 6.8 to show that

$$\Pr\left[\left|\tilde{d}_i - d_i\right| > \epsilon d_i\right] \leq 2e^{-\Upsilon\epsilon^2/3}.$$

The lemma now follows by taking a union bound over i. □

We use the following variant of the Chernoff bound from [Rag88].

Theorem 6.8 (Chernoff Bound). *Let $\alpha_1, \ldots, \alpha_n$ all lie in $[0, \beta]$ and let $X_1, \ldots, X_n$ be independent random variables such that X_i equals α_i with probability p_i and 0 with probability $1 - p_i$. Let $X = \sum_i X_i$ and $\mu = \mathbf{E}[X] = \sum \alpha_i p_i$. Then,*

$$\Pr[X > (1+\epsilon)\mu] < \left(\frac{e^\epsilon}{(1+\epsilon)^{1+\epsilon}} \right)^{\mu/\beta} \quad \text{and} \quad \Pr[X < (1-\epsilon)\mu] < \left(\frac{e^\epsilon}{(1+\epsilon)^{1+\epsilon}} \right)^{\mu/\beta}$$

For $\epsilon < 1$, both of these probabilities are at most $e^{-\mu\epsilon^2/3\beta}$.

We remark that Raghavan [Rag88] proved this theorem with $\beta = 1$; the extension to general $\beta > 0$ follows by re-scaling.

Proof of Theorem 6.1. Let L be the Laplacian of G, A be its adjacency matrix, and D its diagonal matrix of degrees. Let $\widetilde{L}$, $\widetilde{A}$ and $\widetilde{D}$ be the corresponding matrices for $\widetilde{G}$. The matrices L and $\widetilde{L}$ only differ on rows and columns indexed by S. So, if we let $L(S)$ denote the submatrix of L with rows and columns in S, we have

$$\begin{aligned} \left\| D^{-1/2}(L - \widetilde{L})D^{-1/2} \right\| &= \left\| D(S)^{-1/2}(L(S) - \widetilde{L}(S))D(S)^{-1/2} \right\| \\ &\leq \left\| D(S)^{-1/2}(A(S) - \widetilde{A}(S))D(S)^{-1/2} \right\| + \left\| D(S)^{-1/2}(D(S) - \widetilde{D}(S))D(S)^{-1/2} \right\|. \end{aligned}$$

Applying Lemma 6.3 to the first of these terms, while observing

$$\frac{2kn^{1/k}}{\sqrt{\Upsilon}} \leq \frac{4k}{\sqrt{\Upsilon}} = \frac{\epsilon\lambda}{3},$$

we find

$$\Pr\left[\left\| D(S)^{-1/2}(A(S) - \widetilde{A}(S))D(S)^{-1/2} \right\| \geq \frac{\epsilon\lambda}{3} \right] \leq p/3.$$

Applying Lemma 6.7 to the second term, we find

$$\Pr\left[\left\| D(S)^{-1/2}(D(S) - \widetilde{D}(S))D(S)^{-1/2} \right\| \geq \frac{\epsilon\lambda}{3} \right] \leq 2ne^{-\Upsilon(\epsilon\lambda/3)^2/3} < 2ne^{-2k^2} \leq p/3.$$

Thus, with probability at least $1 - 2p/3$,

$$\left\| D^{-1/2}(L - \widetilde{L})D^{-1/2} \right\| \leq \frac{2\epsilon\lambda}{3},$$

in which case Lemma 6.2 tells us that $\widetilde{G}$ is a σ-approximation of G for

$$\sigma = \frac{\lambda}{\lambda - (2/3)\epsilon\lambda} \leq 1 + \epsilon,$$

using $\epsilon \leq 1/2$.

Finally, we use Theorem 6.8 to bound the number of edges in $\widetilde{F}$. For each edge (i, j) in F, let $X_{(i,j)}$ be the indicator random variable for the event that edge (i, j) is chosen to appear in

$\widetilde{F}$. Using d_i to denote the degree of vertex i in $G(S)$, we have

$$\begin{aligned}
\mathbf{E}\left[\sum X_{(i,j)}\right] &= \Upsilon \sum_{(i,j)\in F} \frac{1}{\min(d_i, d_j)} \\
&\leq \Upsilon \sum_{(i,j)\in F} \left(\frac{1}{d_i} + \frac{1}{d_j}\right) \\
&= \Upsilon \sum_{i\in S} \sum_{j:(i,j)\in F} \left(\frac{1}{d_i}\right) \\
&= \Upsilon \left|S\right|.
\end{aligned}$$

One may similarly show that $\mathbf{E}\left[\sum X_{(i,j)}\right] \geq \Upsilon \left|S\right|/2$. Applying Theorem 6.8 with $\epsilon = 1$ (note that here ϵ is the parameter in the statement of Theorem 6.8), we obtain

$$\Pr\left[\sum X_{(i,j)} \geq 2\Upsilon \left|S\right|\right] \leq \left(\frac{e}{4}\right)^{-\Upsilon|S|/2} \leq \left(\frac{e}{4}\right)^{-(8\log_2(3/p))^2} \leq p/3.$$

$\square$

7 Graph Decompositions

In this section, we prove that every graph can be decomposed into components of high conductance, with a relatively small number of edges bridging the components. A similar result was obtained independently by Trevisan [Tre05]. We prove this result for three reasons: first, it enables us to quickly establish the existence of good spectral sparsifiers. Second, our algorithm for building sparsifiers requires a graph decomposition routine which is inspired by the computationally infeasible routine presented in this section[3]. Finally, the analysis of our algorithm relies upon Lemma 7.2, which occupies most of this section. Throughout this section, we will consider an unweighted graph $G = (V, E)$, with $V = \{1, \ldots, n\}$. In the construction of a decomposition of G, we will be concerned with vertex-induced subgraphs of G. However, *when measuring the conductance and volumes of vertices in these vertex-induced subgraphs, we will continue to measure the volume according to the degrees of vertices in the original graph.* For clarity, we define the boundary of a vertex set S with respect to another vertex set B to be

$$\partial_B\left(S\right) = E(S, B - S),$$

we define the conductance of a set S in the subgraph induced by $B \subseteq V$ to be

$$\Phi^G_B\left(S\right) \stackrel{\text{def}}{=} \frac{\left|E(S, B - S)\right|}{\min\left(\mathrm{Vol}\left(S\right), \mathrm{Vol}\left(B - S\right)\right)},$$

and we define

$$\Phi^G_B \stackrel{\text{def}}{=} \min_{S\subset B} \Phi^G_B\left(S\right).$$

[3]The routine `idealDecomp` is infeasible because it requires the solution of an NP-hard problem in step 2. We could construct sparsifiers from a routine that approximately satisfies the guarantees of `idealDecomp`, such as the clustering algorithm of Kannan, Vempala and Vetta [KVV04]. However, their routine could take quadratic time, which is too slow for our purposes.

For convenience, we define $\Phi_B^G(\emptyset) = 1$ and, for $|B| = 1$, $\Phi_B^G = 1$.

We introduce the notation $G\{B\}$ to denote the graph $G(B)$ to which self-loops have been added so that every vertex in $G\{B\}$ has the same degree as in G. For $S \subseteq B$

$$\Phi_{G\{B\}}(S) = \Phi_B^G(S).$$

Because Φ_B^G measures volume by degrees in G and those degrees are higher than in $G(B)$,

$$\Phi_B^G = \Phi_{G\{B\}} \leq \Phi_{G(B)}.$$

So, when we prove lower bounds on Φ_B^G, we obtain lower bounds on $\Phi_{G(B)}$.

7.1 Spectral Decomposition

We define a *decomposition* of G to be a partition of V into sets $(A_1, \ldots, A_k)$, for some k. We say that a decomposition is a *ϕ-decomposition* if $\Phi_{A_i}^G \geq \phi$ for all i. We define the boundary of a decomposition, written $\partial(A_1, \ldots, A_k)$ to be the set of edges between different vertex sets in the partition:

$$\partial(A_1, \ldots, A_k) = E \cap \cup_{i \neq j}(A_i \times A_j).$$

We say that a decomposition $(A_1, \ldots, A_k)$ is a *λ-spectral decomposition* if the smallest non-zero normalized Laplacian eigenvalue of $G(A_i)$ is at least λ, for all i. By Cheeger's inequality (Theorem 4.1), every ϕ-decomposition is a $(\phi^2/2)$-spectral decomposition.

Theorem 7.1. *Let $G = (V, E)$ be a graph and let $m = |E|$. Then, G has a $\left(6 \log_{4/3} 2m\right)^{-1}$-decomposition with $|\partial(A_1, \ldots, A_k)| \leq |E|/2$.*

7.2 Existence of spectral sparsifiers

Before proving Theorem 7.1, we first quickly explain how to use Theorem 7.1 to prove that spectral sparsifiers exist. Given any graph G, apply the theorem to find a decomposition of the graph into components of conductance $\Omega(1/\log n)$, with at most half of the original edges bridging components. Because this decomposition is a $\Omega(1/\log^2 n)$-spectral decomposition, by Theorem 6.1 we may sparsify the graph induced on each component by random sampling. The average degree in the sparsifier for each component will be $O(\log^6 n)$. It remains to sparsify the edges bridging components. If only $\widetilde{\mathcal{O}}(n)$ edges bridge components, then we do not need to sparsify them further. If more edges bridge components, we sparsify them recursively. That is, we treat those edges as a graph in their own right, decompose that graph, sample the edges induced in its components, and so on. As each of these recursive steps reduces the number of edges remaining by at least a factor of two, at most a logarithmic number of recursive steps will be required, and thus the average degree of the sparsifier will be at most $O(\log^7 n)$. The above process also establishes the following decomposition theorem.

Recently, Batson, Spielman and Srivastava [BSS09] have shown that $(1+\epsilon)$-spectral sparsifiers with $O(n/\epsilon^2)$ edges exist.

7.3 The Proof of Theorem 7.1

Theorem 7.1 is not algorithmic. It follows quickly from the following lemma, which says that if the largest set with conductance less than ϕ is small, then the graph induced on the complement has conductance almost ϕ. This lemma is the key component in our proof of Theorem 7.1, and its analog for approximate sparsest cuts (Theorem 8.1) is the key to our algorithm.

Lemma 7.2 (Sparsest Cuts as Certificates)**.** *Let $G = (V, E)$ be a graph and let $\phi \leq 1$. Let $B \subseteq V$ and let $S \subset B$ be a set maximizing* $\mathrm{Vol}(S)$ *among those satisfying*

(C.1) $\mathrm{Vol}(S) \leq \mathrm{Vol}(B)/2$*, and*

(C.2) $\Phi^G_B(S) \leq \phi$*.*

If $\mathrm{Vol}(S) = \alpha \mathrm{Vol}(B)$ *for $\alpha \leq 1/3$, then*

$$\Phi^G_{B-S} \geq \phi\left(\frac{1-3\alpha}{1-\alpha}\right).$$

Proof. Let S be a set of maximum size that satisfies $(C.1)$ and $(C.2)$, let

$$\beta = \frac{1-3\alpha}{1-\alpha},$$

and assume by way of contradiction that $\Phi^G_{B-S} < \phi\beta$. Then, there exists a set $R \subset B - S$ such that

$$\Phi^G_{B-S}(R) < \phi\beta, \text{ and}$$

$$\mathrm{Vol}(R) \leq \frac{1}{2}\mathrm{Vol}(B-S).$$

Let $T = R \cup S$. We will prove

$$\Phi^G_B(T) < \phi$$

and $\mathrm{Vol}(S) \leq \min(\mathrm{Vol}(T), \mathrm{Vol}(B-T))$, contradicting the maximality of S.

We begin by observing that

$$\begin{aligned}|E(T, B-T)| = |E(R \cup S, B - (R \cup S))| &\leq |E(S, B-S)| + |E(R, B-S-R))| \\ &< \phi\mathrm{Vol}(S) + (\phi\beta)\mathrm{Vol}(R). \end{aligned} \tag{12}$$

We divide the rest of our proof into two cases, depending on whether or not $\mathrm{Vol}(T) \leq \mathrm{Vol}(B)/2$. First, consider the case in which $\mathrm{Vol}(T) \leq \mathrm{Vol}(B)/2$. In this case, T provides a contradiction to the maximality of S, as $\mathrm{Vol}(S) < \mathrm{Vol}(T) \leq \mathrm{Vol}(B)/2$, and

$$|E(T, B-T)| < \phi(\mathrm{Vol}(S) + \mathrm{Vol}(R)) = \phi\mathrm{Vol}(T),$$

which implies

$$\Phi^G_B(T) < \phi.$$

In the case $\mathrm{Vol}(T) > \mathrm{Vol}(B)/2$, we will prove that the set $B-T$ contradicts the maximality of S. First, we show

$$\mathrm{Vol}(B-T) > \left(\frac{1-\alpha}{2}\right)\mathrm{Vol}(B), \tag{13}$$

which implies $\mathrm{Vol}(B-T) > \mathrm{Vol}(S)$ because we assume $\alpha \leq 1/3$. To prove (13), compute

$$\begin{aligned}\mathrm{Vol}(T) &= \mathrm{Vol}(S) + \mathrm{Vol}(R) \\ &\leq \mathrm{Vol}(S) + (1/2)(\mathrm{Vol}(B) - \mathrm{Vol}(S)) \\ &= (1/2)\mathrm{Vol}(B) + (1/2)\mathrm{Vol}(S) \\ &= \left(\frac{1+\alpha}{2}\right)\mathrm{Vol}(B).\end{aligned}$$

To upper bound the conductance of T, we compute

$$\begin{aligned}|E(T, B-T)| &< \phi\mathrm{Vol}(S) + (\phi\beta)\mathrm{Vol}(R) \quad \text{(by (12))} \\ &\leq \phi\mathrm{Vol}(S) + (\phi\beta)(\mathrm{Vol}(B) - \mathrm{Vol}(S))/2 \\ &= \phi\mathrm{Vol}(B)(\alpha + \beta(1-\alpha)/2).\end{aligned}$$

So,

$$\Phi^G_B(T) = \frac{|E(T, B-T)|}{\min(\mathrm{Vol}(T), \mathrm{Vol}(B-T))} = \frac{|E(T, B-T)|}{\mathrm{Vol}(B-T)} \leq \frac{\phi\mathrm{Vol}(B)(\alpha + \beta(1-\alpha)/2)}{\mathrm{Vol}(B)(1-\alpha)/2} = \phi,$$

by our choice of β. □

We will prove Theorem 7.1 by proving that the following procedure produces the required decomposition.

Set $\phi = \left(2\log_{4/3}\mathrm{Vol}(V)\right)^{-1}$.
Note that we initially call this algorithm with $B = V$.

`idealDecomp`(B, ϕ)

1. If $\Phi^G_B \geq \phi$, then return B. Otherwise, proceed.
2. Let S be the subset of B maximizing $\mathrm{Vol}(S)$ satisfying (C.1) and (C.2).
3. If $\mathrm{Vol}(S) \leq \mathrm{Vol}(B)/4$, return the decomposition $(B-S,$ `idealDecomp`$(S, \phi))$,
4. else, return the decomposition (`idealDecomp`$(B-S, \phi)$, `idealDecomp`(S, ϕ)).

Proof of Theorem 7.1. To see that the recursive procedure terminates, recall that we have defined $\Phi^G_B = 1$ when $|B| = 1$.

Let $(A_1, \ldots, A_k)$ be the output of `idealDecomp`(V). Lemma 7.2 implies that $\Phi^G_{A_i} \geq \phi/3$ for each i.

To bound the number of edges in $\partial(A_1, \ldots, A_k)$, note that the depth of the recursion is at most $\log_{4/3}\mathrm{Vol}(V)$ and that at most a ϕ fraction of the edges are added to $\partial(A_1, \ldots, A_k)$ at each level of the recursion. So,

$$|\partial(A_1, \ldots, A_k)| \leq |E|\,\phi\log_{4/3}\mathrm{Vol}(V) \leq |E|/2.$$

□

8 Approximate Sparsest Cuts

Unfortunately, it is NP-hard to compute sparsest cuts. So, we cannot directly apply Lemma 7.2 in the design of our algorithm. Instead, we will apply a nearly-linear time algorithm, `ApproxCut`, that computes approximate sparsest cuts that satisfy an analog of Lemma 7.2, stated in Theorem 8.1. Whereas in Lemma 7.2 we proved that if the largest sparse cut is small then its complement has high conductance, here we prove that if the cut output by `ApproxCut` is small, then its complement is contained in a subgraph of high conductance.

The algorithm `ApproxCut` works by repeatedly calling a routine for approximating sparsest cuts, `Partition`, from [ST08a]. On input a graph that contains a sparse cut, with high probability the algorithm `Partition` either finds a large cut or a cut that has high overlap with the sparse cut. We have not been able to find a way to quickly use an algorithm satisfying such a guarantee to certify that the complement of a small cut has high conductance. Kannan, Vempala and Vetta [KVV04] showed that if we applied such an algorithm until it could not find any more cuts then we could obtain such a guarantee. However, such a procedure could require quadratic time, which it too slow for our purposes.

Theorem 8.1 (`ApproxCut`). *Let $\phi, p \in (0,1)$ and let $G = (V,E)$ be a graph with m edges. Let D be the output of* `ApproxCut`(G, ϕ, p)*. Then*

(A.1) $\mathrm{Vol}\,(D) \leq (23/25)\mathrm{Vol}\,(V)$,

(A.2) If $D \neq \emptyset$ then $\Phi_G\,(D) \leq \phi$, and

(A.3) With probability at least $1 - p$, either

- *(A.3.a)* $\mathrm{Vol}\,(D) \geq (1/29)\mathrm{Vol}\,(V)$*, or*
- *(A.3.b) there exists a set $W \supseteq V - D$ for which $\Phi_W^G \geq f_2(\phi)$, where*

$$f_2(\phi) \stackrel{\text{def}}{=} \frac{c_2 \phi^2}{\log^4 m}, \tag{14}$$

for some absolute constant c_2.

Moreover, the expected running time of `ApproxCut` *is* $O\left(\phi^{-4} m \log^9 m \log(1/p)\right)$.

The code for `ApproxCut` follows. It relies on a routine called `Partition2` which in turn relies on a routine called `Partition` from [ST08a]. While one could easily combine the routines `ApproxCut` and `Partition2`, their separation simplifies our analysis. The algorithm `Partition2` is very simple: it just calls `Partition` repeatedly and collects the cuts it produces until they contain at least 1/5 of the volume of the graph or until it has made enough calls. The algorithm `ApproxCut` is similar: it calls `Partition2` in the same way that `Partition2` calls `Partition`.

$D = \texttt{ApproxCut}(G, \phi, p)$, where G is a graph, $\phi, p, \in (0, 1)$.

(0) Set $V_0 = V$ and $j = 0$.

(1) Set $r = \lceil \log_2(m) \rceil$ and $\epsilon = \min(1/2r, 1/5)$.

(2) While $j < r$ and $\mathrm{Vol}\,(V_j) \geq (4/5)\mathrm{Vol}\,(V)$,

    (a) Set $j = j + 1$.
    (b) Set $D_j = \texttt{Partition2}(G\{V_{j-1}\}, (2/23)\phi, p/2r, \epsilon)$
    (c) Set $V_j = V_{j-1} - D_j$.

(3) Set $D = D_1 \cup \cdots \cup D_j$.

8.1 Partitioning in Nearly-Linear-Time

$D = \texttt{Partition2}(G, \theta, p, \epsilon)$, where G is a graph, $\theta, p, \in (0, 1)$ and $\epsilon \in (0, 1)$.

(0) Set $W_0 = V$ and $j = 0$. Set $r = \lceil \log_2(1/\epsilon) \rceil$.

(1) While $j < r$ and $\mathrm{Vol}\,(W_j) \geq (4/5)\mathrm{Vol}\,(V)$,

    (a) Set $j = j + 1$.
    (b) Set $D_j = \texttt{Partition}(G\{W_{j-1}\}, \theta/9, p/r)$
    (c) Set $W_j = W_{j-1} - D_j$.

(2) Set $D = D_1 \cup \cdots \cup D_j$.

The algorithm `Partition` from [ST08a], satisfies the following theorem (see [ST08a, Theorem 3.2])

Theorem 8.2 (`Partition`)**.** *Let D be the output of* $\texttt{Partition}(G, \tau, p)$*, where G is a graph and $\tau, p \in (0, 1)$. Then*

(P.1) $\mathrm{Vol}\,(D) \leq (7/8)\mathrm{Vol}\,(V)$,

(P.2) If $D \neq \emptyset$ then $\Phi_G\,(D) \leq \tau$, and

(P.3) For some absolute constant c_1 and

$$f_1(\tau) \stackrel{\text{def}}{=} \frac{c_1 \tau^2}{\log^3 m},$$

*for **every** set S satisfying*

$$\mathrm{Vol}\,(S) \leq \mathrm{Vol}\,(V)/2 \quad \textit{and} \quad \Phi_G\,(S) \leq f_1(\tau), \tag{15}$$

with probability at least $1 - p$ either

(P.3.a) $\mathrm{Vol}(D) \geq (1/4)\mathrm{Vol}(V)$, *or*

(P.3.b) $\mathrm{Vol}(S \cap D) \geq \mathrm{Vol}(S)/2$.

Moreover, the expected running time of `Partition` *is* $O\left(\tau^{-4} m \log^7 m \log(1/p)\right)$.

If either $(P.3.a)$ or $(P.3.b)$ occur for a set S satisfying (15), we say that `Partition` **succeeds** for S. Otherwise, we say that it **fails**.

One can view condition $(A.3)$ in Theorem 8.1 as reversing the quantifiers in condition $(P.3)$ in Theorem 8.2. Theorem 8.2 says that for every set S of low conductance there is a good probability that a substantial portion of S is removed. On the other hand, Theorem 8.1 says that with high probability all sets of low conductance will be removed.

The algorithm `Partition2` satisfies a guarantee similar to that of `Partition`, but it strengthens condition $(P.3.b)$.

Lemma 8.3 (`Partition2`)**.** *Let D be the output of* `Partition2`(G, θ, p, ϵ)*, where G is a graph, $\theta, p \in (0,1)$ and $\epsilon \in (0,1)$. Then*

(Q.1) $\mathrm{Vol}(D) \leq (9/10)\mathrm{Vol}(V)$,

(Q.2) If $D \neq \emptyset$ then $\Phi_G(D) \leq \theta$, and

*(Q.3) For **every** set S satisfying*

$$\mathrm{Vol}(S) \leq \mathrm{Vol}(V)/2 \quad \text{and} \quad \Phi_G(S) \leq f_1(\theta/9), \tag{16}$$

with probability at least $1-p$, either

(Q.3.a) $\mathrm{Vol}(D) \geq (1/5)\mathrm{Vol}(V)$, *or*

(Q.3.b) $\mathrm{Vol}(S \cap D) \geq (1-\delta)\mathrm{Vol}(S)$, *where* $\delta = \max(\epsilon, \Phi_G(S)/f_1(\theta/9))$.

Moreover, the expected running time of `Partition2` *is* $O\left(\theta^{-4} m \log^7 m \log(1/\epsilon) \log(\log(1/\epsilon)/p)\right)$.

If either $(Q.3.a)$ or $(Q.3.b)$ occur for a set S satisfying (16), we say that `Partition2` **succeeds** for S. Otherwise, we say that it **fails**.

The proof of this lemma is routine, given Theorem 8.2.

Proof. Let j^* be such that $D = D_1 \cup \cdots \cup D_{j^*}$. To prove (Q.1), let $\nu = \mathrm{Vol}((D_1 \cup \cdots \cup D_{j^*-1}))/\mathrm{Vol}(V)$. As $\mathrm{Vol}(W_{j^*-1}) \geq (4/5)\mathrm{Vol}(V)$, $\nu \leq 1/5$. By $(P.1)$, $\mathrm{Vol}(D_{j^*}) \leq (7/8)\mathrm{Vol}(W_{j^*-1})$, so

$$\mathrm{Vol}(D_1 \cup \cdots \cup D_{j^*}) \leq \mathrm{Vol}(V)(\nu + (7/8)(1-\nu)) \leq \mathrm{Vol}(V)((1/5) + (7/8)(4/5)) = (9/10)\mathrm{Vol}(V).$$

To establish (Q.2), we first compute

$$
\begin{aligned}
|E(D, V-D)| &= \sum_{i=1}^{j^*} |E(D_i, V-D)| \\
&\leq \sum_{i=1}^{j^*} |E(D_i, W_{i-1}-D_i)| \\
&\leq \sum_{i=1}^{j^*} (\theta/9) \min\left(\mathrm{Vol}\left(D_i\right), \mathrm{Vol}\left(W_{i-1}-D_i\right)\right) \quad \text{(by (P.2) and line 1b of } \texttt{Partition2}\text{)} \\
&\leq \sum_{i=1}^{j^*} (\theta/9) \mathrm{Vol}\left(D_i\right) \\
&= (\theta/9) \mathrm{Vol}\left(D\right).
\end{aligned}
$$

So, if $\mathrm{Vol}\left(D\right) \leq \mathrm{Vol}\left(V\right)/2$, then $\Phi_G\left(D\right) \leq \theta/9$. On the other hand, we established above that $\mathrm{Vol}\left(D\right) \leq (9/10)\mathrm{Vol}\left(V\right)$, from which it follows that

$$
\mathrm{Vol}\left(V-D\right) \geq (1/10)\mathrm{Vol}\left(V\right) \geq (1/10)(10/9)\mathrm{Vol}\left(D\right) = (1/9)\mathrm{Vol}\left(D\right).
$$

So,

$$
\Phi_G\left(D\right) = \frac{|E(D, V-D)|}{\min\left(\mathrm{Vol}\left(D\right), \mathrm{Vol}\left(V-D\right)\right)} \leq 9\frac{|E(D, V-D)|}{\mathrm{Vol}\left(D\right)} \leq \theta.
$$

To prove (Q.3), let S be a set satisfying (16), and let $S_j = S \cap W_j$. From Theorem 8.2, we know that with probability at least $1 - p/r$,

$$
\mathrm{Vol}\left(S_1\right) \leq (1/2)\mathrm{Vol}\left(S_0\right). \tag{17}
$$

We need to prove that with probability at least $1-p$, either $\mathrm{Vol}\left(W_{j^*}\right) \leq (4/5)\mathrm{Vol}\left(V\right)$ or $\mathrm{Vol}\left(S_{j^*}\right) \leq \delta\mathrm{Vol}\left(S\right)$. If neither of these inequalities hold, then

$$
j^* = r, \quad \mathrm{Vol}\left(W_r\right) \geq (4/5)\mathrm{Vol}\left(V\right), \quad \text{and} \quad \mathrm{Vol}\left(S_r\right) > \delta\mathrm{Vol}\left(S\right) \geq \epsilon\mathrm{Vol}\left(S\right),
$$

where we recall $r = \lceil \log_2(1/\epsilon) \rceil$. So, there must exist a j for which $\mathrm{Vol}\left(S_{j+1}\right) \geq (1/2)\mathrm{Vol}\left(S_j\right)$. If S_j satisfied condition (16) in $G\{V_j\}$ this would imply that `Partition` failed for S_j. We already know this is unlikely for $j = 0$. To show it is unlikely for $j \geq 1$, we prove that S_j does satisfy condition (16) in $G\{V_j\}$. Assuming (17),

$$
\begin{aligned}
\Phi_{G\{W_j\}}\left(S_j\right) = \Phi^G_{W_j}\left(S_j\right) = \frac{\left|\partial_{W_j}\left(S_j\right)\right|}{\min\left(\mathrm{Vol}\left(S_j\right), \mathrm{Vol}\left(W_j - S_j\right)\right)} &= \frac{\left|\partial_{W_j}\left(S_j\right)\right|}{\mathrm{Vol}\left(S_j\right)} \leq \frac{\left|\partial_V\left(S\right)\right|}{\mathrm{Vol}\left(S_r\right)} \\
&\leq \frac{\left|\partial_V\left(S\right)\right|}{\delta\mathrm{Vol}\left(S\right)} = (1/\delta)\Phi_G\left(S\right) \leq f_1(\theta/9),
\end{aligned}
$$

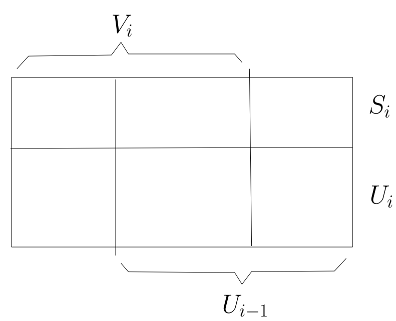
where the third equality follows from the assumption $\mathrm{Vol}\left(S_1\right) \leq (1/2)\mathrm{Vol}\left(S_0\right) \leq (1/4)\mathrm{Vol}\left(V\right)$ and the last inequality follows from the definition $\delta = \max\left(\epsilon, \Phi_G\left(S\right)/f_1(\theta/9)\right)$. So, S_j satisfies conditions (15) with $\tau = \theta/9$, but `Partition` fails for S_j. As there are at most r sets S_j, this happens for one of them with probability at most $r(p/r) = p$.

Finally, the bound on the expected running time of `Partition2` is immediate from the bound on the running time of `Partition`. $\square$

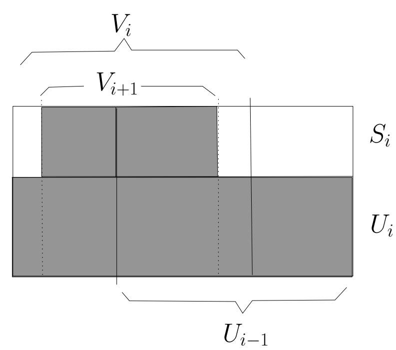
8.2 Proof of Theorem 8.1

The rest of this section is devoted to the proof of Theorem 8.1, with all but one line devoted to part (A.3). Our goal is to prove the existence of a set of vertices W of high conductance that contains all the vertices not cut out by `ApproxCut`. We will construct this set W in stages. Recall that $V_i = V - D_1 \cup \cdots \cup D_i$ is the set of vertices that are not removed by the first i cuts. In stage i, we will express W_i, a superset of V_i, as a set of high conductance U_{i-1} plus some vertices in V_i. We will show that in each stage the volume of the vertices that are not in the set of high conductance shrinks by at least a factor of 2.

We do this by letting S_i be the biggest set of conductance at most σ_i in W_i, where σ_i is a factor $(1-2\epsilon)$ smaller than the conductance of U_{i-1}. We then show that at least a 2ϵ fraction of the volume of S_i lies outside U_{i-1} and thus inside V_i. From Lemma 7.2 we know that $U_i \stackrel{\text{def}}{=} W_i - S_i$ has high conductance. We will use Lemma 8.3 to show that at most an ϵ fraction of S_i appears in V_{i+1}. So, the volume of S_i that remains inside V_{i+1} will be at most half the volume of V_i that is not in U_{i-1}. We then set $W_{i+1} = U_i \cup (S_i \cap V_{i+1})$, and proceed with our induction. Eventually, we will arrive at an i for which either W_i has high conductance or enough volume has been removed from V_i.



The subsets of W_i. Not drawn to scale.



The shaded portion is W_{i+1}. It equals $U_i \cup V_{i+1}$, and so can be viewed as the union of the set of vertices maintained by the algorithm with the high-conductance set we know exists.

Formally, we set

$$W_0 = V_0 = V \quad \text{and} \quad \sigma_0 = \epsilon f_1(\phi/104).$$

We then construct sets S_i, U_i and W_i by the following inductive procedure.

1. Set $i = 0$.

2. While $i \le r$ and W_i is defined,

   a. If W_i contains a set S_i such that

   $$\mathrm{Vol}\left(S_i\right) \le (1/2)\mathrm{Vol}\left(W_i\right) \quad \text{and} \quad \Phi^G_{W_i}\left(S_i\right) \le \sigma_i,$$

   set S_i to be such a set of maximum size.

If $\mathrm{Vol}(S_i) \geq (2/17)\mathrm{Vol}(V)$, stop the procedure and leave W_{i+1} undefined.
If there is no such set, set $S_i = \emptyset$, set $U_i = W_i$, stop the procedure and leave W_{i+1} undefined.

b. Set $U_i = W_i - S_i$.

c. Set $\theta_i = \left(1 - 3\frac{\mathrm{Vol}(S_i)}{\mathrm{Vol}(W_i)}\right)\sigma_i$.

d. Set $\sigma_{i+1} = (1 - 2\epsilon)\theta_i$.

e. Set $W_{i+1} = U_i \cup (S_i \cap V_{i+1})$.

f. Set $i = i + 1$.

3. Set $W = W_i$ where i is the last index for which W_i is defined.

Note that there may be many choices for a set S_i. Once a choice is made, it must be fixed for the rest of the procedure so that we can reason about it using Lemma 8.3.

We will prove that if some set S_i has volume greater than $(2/17)\mathrm{Vol}(V)$, then with high probability `ApproxCut` will return a large cut D, and hence part (A.3.a) is satisfied. Thus, we will be mainly concerned with the case in which this does not happen. In this case, we will prove that θ_i is not too much less than σ_0, and so the set U_i has high conductance. If the procedure stops because S_i is empty, then $W_i = U_i$ is the set of high conductance we seek. We will prove that for some $i \leq r$ probably either S_i is empty, $\mathrm{Vol}(S_i) \geq (2/17)\mathrm{Vol}(V)$ or $\mathrm{Vol}(V_i) \leq (16/17)\mathrm{Vol}(V)$.

Claim 8.4. *For all i such that W_{i+1} is defined,*

$$V_{i+1} \subseteq W_{i+1} \subseteq W_i.$$

Proof. We prove this by induction on i. For $i = 0$, we know that $V_i = W_i$. As $W_i = U_i \cup S_i$ and the algorithm ensures $V_{i+1} \subseteq V_i$,

$$V_{i+1} \subseteq V_i \subseteq W_i = U_i \cup S_i.$$

Thus,

$$V_{i+1} \subseteq U_i \cup (S_i \cap V_{i+1}) = W_{i+1} \subseteq U_i \cup S_i = W_i.$$

□

Claim 8.5. *For all i such that U_i is defined*

$$\Phi^G_{U_i} \geq \theta_i.$$

Proof. Follows immediately from Lemma 7.2 and the definitions of S_i and θ_i. □

Lemma 8.6. *If*

(a) $\mathrm{Vol}(S_i) \leq (2/17)\mathrm{Vol}(V)$, *and*

(b) $\mathrm{Vol}(V_{i-1}) \geq (16/17)\mathrm{Vol}(V)$, *then*

then

$$\mathrm{Vol}\left(S_i \cap (S_{i-1} \cap V_i)\right) \geq 2\epsilon \mathrm{Vol}\left(S_i\right).$$

Proof. This lemma follows easily from the definitions of the sets S_i, U_i and V_i. As $V_{i-1} \subseteq W_{i-1}$ and $\mathrm{Vol}\left(U_{i-1}\right) \geq (1/2)\mathrm{Vol}\left(W_{i-1}\right)$,

$$\mathrm{Vol}\left(U_{i-1}\right) \geq (1/2)\mathrm{Vol}\left(V_{i-1}\right) \geq (8/17)\mathrm{Vol}\left(V\right) \geq 4\mathrm{Vol}\left(S_i\right).$$

So, we may apply Claim 8.5 to show

$$\left|\partial_{U_{i-1}}\left(S_i\right)\right| \geq \left|\partial_{U_{i-1}}\left(S_i \cap U_{i-1}\right)\right| \geq \theta_{i-1}\mathrm{Vol}\left(S_i \cap U_{i-1}\right).$$

On the other hand,

$$\left|\partial_{U_{i-1}}\left(S_i\right)\right| \leq \left|\partial_{W_i}\left(S_i\right)\right| \leq \sigma_i \mathrm{Vol}\left(S_i\right) = (1-2\epsilon)\theta_{i-1}\mathrm{Vol}\left(S_i\right).$$

Combining these two inequalities yields

$$\theta_{i-1}\mathrm{Vol}\left(S_i \cap U_{i-1}\right) \leq (1-2\epsilon)\theta_{i-1}\mathrm{Vol}\left(S_i\right)$$

and

$$\mathrm{Vol}\left(S_i \cap U_{i-1}\right) \leq (1-2\epsilon)\mathrm{Vol}\left(S_i\right).$$

As

$$S_i \subseteq W_i = U_{i-1} \cup (S_{i-1} \cap V_i),$$

we may conclude

$$\mathrm{Vol}\left(S_i \cap (S_{i-1} \cap V_i)\right) \geq 2\epsilon \mathrm{Vol}\left(S_i\right).$$

□

We now show that if at most an ϵ fraction of each S_i appears in V_{i+1}, then the sets $S_i \cap V_{i+1}$ shrink to the point of vanishing.

Lemma 8.7. *If all defined S_i and V_i satisfy*

(a) $\mathrm{Vol}\left(S_i\right) \leq (2/17)\mathrm{Vol}\left(V\right)$,

(b) $\mathrm{Vol}\left(V_i\right) \geq (16/17)\mathrm{Vol}\left(V\right)$, *and*

(c) $\mathrm{Vol}\left(S_i \cap V_{i+1}\right) \leq \epsilon \mathrm{Vol}\left(S_i\right)$,

then for all $i \geq 1$ for which S_i is defined,

$$\mathrm{Vol}\left(S_i \cap V_{i+1}\right) \leq (1/2)\mathrm{Vol}\left(S_{i-1} \cap V_i\right),$$

and

$$\mathrm{Vol}\left(S_i\right) \leq (1/2)\mathrm{Vol}\left(S_{i-1}\right).$$

In particular, the set S_r is empty if it is defined.

Proof. Lemma 8.6 tells us that

$$\epsilon \mathrm{Vol}\left(S_i\right) \leq (1/2)\mathrm{Vol}\left(S_i \cap (S_{i-1} \cap V_i)\right) \leq (1/2)\mathrm{Vol}\left(S_{i-1} \cap V_i\right).$$

Combining this inequality with (c) yields

$$\mathrm{Vol}\left(S_i \cap V_{i+1}\right) \leq (1/2)\mathrm{Vol}\left(S_{i-1} \cap V_i\right).$$

Similarly, we may conclude from Lemma 8.6 that

$$\epsilon \mathrm{Vol}\left(S_{i+1}\right) \leq (1/2)\mathrm{Vol}\left(S_i \cap V_{i+1}\right),$$

which when combined with (c) yields

$$\epsilon \mathrm{Vol}\left(S_{i+1}\right) \leq (1/2)\epsilon \mathrm{Vol}\left(S_i\right),$$

from which the second part of the lemma follows.

For S_i to be defined, we must have $\mathrm{Vol}\left(S_0\right) \leq (2/17)\mathrm{Vol}\left(V\right)$; so,

$$\mathrm{Vol}\left(S_r\right) \leq (1/2)^r \mathrm{Vol}\left(S_0\right) \leq (1/2)^{\lceil \log_2 \mathrm{Vol}(V)/2 \rceil}(2/17)\mathrm{Vol}\left(V\right) \leq \frac{2}{\mathrm{Vol}\left(V\right)}(2/17)\mathrm{Vol}\left(V\right) < 1.$$

We conclude that the set S_r must be empty if it is defined. $\square$

This geometric shrinking of the volumes of the sets S_i allows us to prove a lower bound on θ_i.

Lemma 8.8. *Under the conditions of Lemma 8.7,*

$$\theta_i \geq \frac{c_2 \phi^2}{\log^4 m},$$

for some absolute constant c_2.

Proof. We have

$$\theta_i = \sigma_0 (1 - 2\epsilon)^{i-1} \prod_{j=0}^{i} \left(1 - \frac{3\mathrm{Vol}\left(S_j\right)}{\mathrm{Vol}\left(W_j\right)}\right).$$

As $i \leq r$ and $\epsilon = \min(1/5, 1/2r)$, we have

$$(1 - 2\epsilon)^{i-1} \geq 1/e.$$

To analyze the other product, we apply Lemma 8.7 to prove

$$\sum_{j=0}^{i} \mathrm{Vol}\left(S_j\right) \leq 2\mathrm{Vol}\left(S_0\right),$$

and so

$$\prod_{j=0}^{i}\left(1-\frac{3\mathrm{Vol}\left(S_j\right)}{\mathrm{Vol}\left(W_j\right)}\right) \geq 1-\sum_{i=0}^{r}\frac{3\mathrm{Vol}\left(S_i\right)}{(16/17)\mathrm{Vol}\left(V\right)}$$
$$\geq 1-\frac{2\cdot 3\cdot 17}{16}\frac{\mathrm{Vol}\left(S_0\right)}{\mathrm{Vol}\left(V\right)},$$
$$\geq 1-\frac{2\cdot 3\cdot 17}{16}\frac{2}{17}$$
$$=\frac{1}{4}.$$

Thus,

$$\theta_i \geq \frac{\sigma_0}{4e} \geq \frac{\epsilon f_1(\phi/104)}{4e} \geq \frac{c_1\phi^2}{4e(104)^2\lceil\log m\rceil\log^3 m} \geq \frac{c_2\phi^2}{\log^4 m},$$

for some constant c_2. □

To prove that condition (c) of Lemma 8.7 is probably satisfied, we will consider two cases. First, if $\mathrm{Vol}\left(S_i \cap V_i\right) \leq \epsilon\mathrm{Vol}\left(S_i\right)$ then (c) is trivially satisfied as $V_{i+1} \subseteq V_i$. On the other hand, if $\mathrm{Vol}\left(S_i \cap V_i\right) \geq \epsilon\mathrm{Vol}\left(S_i\right)$, then we will show that $S_i \cap V_i$ satisfies conditions (16) in $G\{V_i\}$, and so with high probability the cut D_{i+1} made by `Partition2` removes enough of S_i.

Lemma 8.9. *If*

(a) $\mathrm{Vol}\left(S_i\right) \leq (2/17)\mathrm{Vol}\left(V\right)$,

(b) $\mathrm{Vol}\left(V_i\right) \geq (16/17)\mathrm{Vol}\left(V\right)$, *and*

(c) $\mathrm{Vol}\left(S_i \cap V_i\right) \geq \epsilon\mathrm{Vol}\left(S_i\right)$,

then

$$\Phi_{G\{V_i\}}\left(S_i \cap V_i\right) \leq \frac{\epsilon}{\delta}f_1(\phi/104),$$

where $\delta = \mathrm{Vol}\left(S_i \cap V_i\right)/\mathrm{Vol}\left(S_i\right)$. *If, in addition*

$$\mathrm{Vol}\left(S_i \cap V_{i+1}\right) \leq \frac{\epsilon}{\delta}\mathrm{Vol}\left(S_i \cap V_i\right),$$

then

$$\mathrm{Vol}\left(S_i \cap V_{i+1}\right) \leq \epsilon\mathrm{Vol}\left(S_i\right).$$

Proof. By Claim 8.10,

$$\left|\partial_{V_i}\left(S_i \cap V_i\right)\right| \leq \left|\partial_{W_i}\left(S_i\right)\right|.$$

Set $\delta = \mathrm{Vol}\left(S_i \cap V_i\right)/\mathrm{Vol}\left(S_i\right)$. Assumption (c) tells us that $\delta \geq \epsilon$. As $\mathrm{Vol}\left(S_i\right) \leq (1/2)\mathrm{Vol}\left(V_i\right)$,

$$\Phi_{G\{V_i\}}\left(S_i \cap V_i\right) = \frac{\left|\partial_{V_i}\left(S_i \cap V_i\right)\right|}{\mathrm{Vol}\left(S_i \cap V_i\right)} \leq \frac{\left|\partial_{W_i}\left(S_i\right)\right|}{\delta\mathrm{Vol}\left(S_i\right)} = \frac{1}{\delta}\Phi_{G\{W_i\}}\left(S_i\right) \leq \frac{\sigma_i}{\delta} = \frac{\sigma_i}{\epsilon}\frac{\epsilon}{\delta} \leq \frac{\sigma_0}{\epsilon}\frac{\epsilon}{\delta} = \frac{\epsilon}{\delta}f_1(\phi/104).$$

The last part of the lemma is trivial. □

Claim 8.10.

$$\partial_{V_i}(S_i \cap V_i) \subseteq \partial_{W_i}(S_i).$$

Proof.

$$\partial_{V_i}(S_i \cap V_i) = E(S_i \cap V_i, V_i - (S_i \cap V_i)) \subseteq E(S_i, V_i - (S_i \cap V_i)) \subseteq E(S_i, W_i - (S_i \cap W_i)) = \partial_{W_i}(S_i).$$

□

We now show that if $\mathrm{Vol}(S_i) \geq (2/17)\mathrm{Vol}(V)$, then in the ith iteration `Partition2` will probably remove a large portion of the graph. If $\mathrm{Vol}(S_i \cap V_i) \leq (1/2)\mathrm{Vol}(V_i)$ we will argue that $S_i \cap V_i$ satisfies condition (16) in $G\{V_i\}$. Otherwise, will argue that $V_i - S_i \cap V_i$ does.

Lemma 8.11. *If*

(a) $\mathrm{Vol}(V_i) \geq (16/17)\mathrm{Vol}(V)$,

(b) $\mathrm{Vol}(S_i) \geq (2/17)\mathrm{Vol}(V)$, *and*

(c) $\mathrm{Vol}(S_i \cap V_i) \leq (1/2)\mathrm{Vol}(V_i)$,

then

$$\Phi_{G\{V_i\}}(S_i \cap V_i) \leq 2\epsilon f_1(\phi/104).$$

Moreover, if $\mathrm{Vol}(S_i \cap V_i \cap D_{i+1}) \geq (1-2\epsilon)\mathrm{Vol}(S_i \cap V_i)$ *then*

$$\mathrm{Vol}(D_{i+1}) \geq (1/29)\mathrm{Vol}(V).$$

Proof. We first lower-bound the volume of the intersection of S_i with V_i by

$$\mathrm{Vol}(S_i \cap V_i) \geq \mathrm{Vol}(S_i) - (\mathrm{Vol}(V) - \mathrm{Vol}(V_i)) \geq \mathrm{Vol}(S_i) - (1/17)\mathrm{Vol}(V) \geq (1/2)\mathrm{Vol}(S_i).$$

We then apply Claim 8.10 to show

$$\Phi_{G\{V_i\}}(S_i \cap V_i) = \frac{|\partial_{V_i}(S_i \cap V_i)|}{\mathrm{Vol}(S_i \cap V_i)} \leq \frac{|\partial_{W_i}(S_i)|}{(1/2)\mathrm{Vol}(S_i)} \leq 2\sigma_i \leq 2\epsilon f_1(\phi/104).$$

The last part of the lemma follows from $\mathrm{Vol}(S_i \cap V_i) \geq (1/17)\mathrm{Vol}(V)$ and $\epsilon \leq 1/5$. □

Lemma 8.12. *If*

(a) $\mathrm{Vol}(V_i) \geq (16/17)\mathrm{Vol}(V)$ *and*

(b) $\mathrm{Vol}(S_i \cap V_i) \geq (1/2)\mathrm{Vol}(V_i)$,

then

$$\Phi_{G\{V_i\}}(S_i \cap V_i) \leq 2\epsilon f_1(\phi/104).$$

Moreover, if $\mathrm{Vol}((V_i - (S_i \cap V_i)) \cap D_{i+1}) \geq (1-\epsilon)\mathrm{Vol}((V_i - (S_i \cap V_i)))$ *then*

$$\mathrm{Vol}(D_{i+1}) \geq (3/16)\mathrm{Vol}(V).$$

Proof. As $\mathrm{Vol}\,(S_i) \leq (1/2)\mathrm{Vol}\,(W_i) \leq (1/2)\mathrm{Vol}\,(V)$ and $\mathrm{Vol}\,(V_i - S_i \cap V_i) \geq \mathrm{Vol}\,(V_i) - \mathrm{Vol}\,(S_i) \geq (15/34)\mathrm{Vol}\,(V)$,

$$\mathrm{Vol}\,(V_i - S_i \cap V_i) \geq (15/17)\mathrm{Vol}\,(S_i)\,.$$

So, by Claim 8.10,

$$\Phi_{G\{V_i\}}\,(V_i - (V_i \cap S_i)) = \frac{|\partial_{V_i}\,(S_i \cap V_i)|}{\mathrm{Vol}\,(V_i - (V_i \cap S_i))} \leq (17/15)\frac{|\partial_{W_i}\,(S_i)|}{\mathrm{Vol}\,(S_i)} \leq (17/15)\sigma_0 \leq 2\epsilon f_1(\phi/104).$$

The last part now follows from

$$\mathrm{Vol}\,(V_i - S_i \cap V_i) \geq (15/17)\mathrm{Vol}\,(S_i) \geq \frac{15}{17}\frac{1}{2}\mathrm{Vol}\,(V_i) \geq (5/16)\mathrm{Vol}\,(V)$$

and $\epsilon \leq 1/5$. □

Proof of Theorem 8.1. The proofs of (A.1) and (A.2) are similar to the proofs of (Q.1) and (Q.2).

To prove (A.3), we will assume that for each set S_i that satisfies conditions (16) in $G\{V_i\}$ the call to `Partition2` succeeds and that the same holds for all sets $V_i - S_i$ that satisfy conditions (16) in $G\{V_i\}$. As this assumption involves at most $2r$ sets, by Lemma 8.3 it holds with probability at least $1 - 2r(p/2r) = 1 - p$.

If there is an i for which $\mathrm{Vol}\,(V_i) < (16/17)\mathrm{Vol}\,(V)$, then $\mathrm{Vol}\,(D) \geq (1/17)V$ and condition $(A.3.a)$ is satisfied. So, we assume that $\mathrm{Vol}\,(V_i) \geq (16/17)\mathrm{Vol}\,(V)$ for the rest of the proof.

Observe that the algorithm `ApproxCut` calls `Partition2` with

$$\theta = (2/23)\phi,$$

and that

$$\phi/104 < \theta/9.$$

So, if $\mathrm{Vol}\,(S_i \cap V_i) \leq \mathrm{Vol}\,(V_i)\,/2$ and

$$\Phi_{G\{V_i\}}\,(S_i) \leq f_1(\phi/104),$$

then S_i satisfies the conditions (16) in $G\{V_i\}$.

If there is an i for which $\mathrm{Vol}\,(S_i) \geq (2/17)\mathrm{Vol}\,(V)$, then by Lemmas 8.11 and 8.12 either $S_i \cap V_i$ or $V_i - (S_i \cap V_i)$ satisfies conditions (16) in $G\{V_i\}$ and the success of the call to `Partition2` implies

$$\mathrm{Vol}\,(D) \geq (1/29)\mathrm{Vol}\,(V)\,.$$

So, for the rest of the proof we may assume $\mathrm{Vol}\,(S_i) \leq (2/17)\mathrm{Vol}\,(V)$. In this case we may show that

$$\mathrm{Vol}\,(S_i \cap V_{i+1}) \leq \epsilon \mathrm{Vol}\,(S_i) \tag{18}$$

as follows. If $\mathrm{Vol}\,(S_i \cap V_i) \leq \epsilon\mathrm{Vol}\,(S_i)$ then (18) trivially holds. Otherwise, Lemma 8.9 tells us that S_i satisfies conditions (16) in $G\{V_i\}$ and that the success of the call to `Partition2` guarantees (18).

We may now apply Lemma 8.7 to show that S_r is empty if it is defined. So, there is an i for which $W_i = U_i$ and by Claim 8.5 and Lemma 8.8

$$\Phi^G_{W_i} \geq \frac{c_2\phi^2}{\log^4 m}.$$

as $V - D = V_r \subseteq V_i \subseteq W_i$, the set $W = W_i$ satisfies (A.3.b). □

9 Sparsifying Unweighted Graphs

We now show how to use the algorithms `ApproxCut` and `Sample` to sparsify unweighted graphs. More precisely, we treat every edge in an unweighted graph as an edge of weight 1. The algorithm `UnwtedSparsify` follows the outline described in Section 7.2. Its main subroutine `PartitionAndSample` calls `ApproxCut` to partition the graph. Whenever `ApproxCut` returns a small cut, we know that the complement is contained in a subgraph of large conductance. In this case, `PartitionAndSample` calls `Sample` to sparsify the large part. Whenever the cut returned by `ApproxCut` is large, `PartitionAndSample` recursively acts on the cut and its complement so that it eventually partitions and samples both. The output of `PartitionAndSample` is the result of running `Sample` on the graphs induced on the vertex sets of a decomposition of the original graph. The main routine `UnwtedSparsify` calls `PartitionAndSample` and then acts recursively to sparsify the edges that go between the parts of the decomposition produced by `PartitionAndSample`.

$\widetilde{G} = \texttt{UnwtedSparsify}(G, \epsilon, p)$

1. If $\mathrm{Vol}\,(V) \leq c_3 \epsilon^{-2} n \log^{30}(n/p)$, return G (where c_3 is set in the proof of Lemma 9.1).
2. Set $\phi = \left(2 \log_{29/28} \mathrm{Vol}\,(V)\right)^{-1}$, $\hat{p} = p/6n \log_2 n$, and $\hat{\epsilon} = \frac{\epsilon (\ln 2)^2}{(1 + 2\log_{29/28} n)(2 \log n)}$.
3. Set $(\widetilde{G}_1, \ldots, \widetilde{G}_k) = \texttt{PartitionAndSample}(G, \phi, \hat{\epsilon}, \hat{p})$.
4. Let $V_1, \ldots, V_k$ be the vertex sets of $\widetilde{G}_1, \ldots, \widetilde{G}_k$, respectively, and let G_0 be the graph with vertex set V and edge set $\partial\,(V_1, \ldots, V_k)$.
5. Set $\widetilde{G}_0 = \texttt{UnwtedSparsify}(G_0, \epsilon, p)$.
6. Set $\widetilde{G} = \sum_{i=0}^{k} \widetilde{G}_i$.

$(\widetilde{G}_1, \ldots, \widetilde{G}_k) = \texttt{PartitionAndSample}(G = (V, E), \phi, \hat{\epsilon}, \hat{p})$

0. Set $\lambda = f_2(\phi)^2/2$, where f_2 is defined in (14).
1. Set $D = \texttt{ApproxCut}(G, \phi, \hat{p})$.
2. If $D = \emptyset$, return $\widetilde{G}_1 = \texttt{Sample}(G, \hat{\epsilon}, \hat{p}, \lambda)$.
3. Else, if $\mathrm{Vol}\,(D) \leq (1/29)\mathrm{Vol}\,(V)$
   a. Set $\widetilde{G}_1 = \texttt{Sample}(G(V - D), \hat{\epsilon}, \hat{p}, \lambda)$
   b. Return $(\widetilde{G}_1, \texttt{PartitionAndSample}(G(D), \phi, \hat{\epsilon}, \hat{p}))$.
4. Else,
   a. Set $\widetilde{H}_1, \ldots, \widetilde{H}_k = \texttt{PartitionAndSample}(G(V - D), \phi, \hat{\epsilon}, \hat{p})$.
   b. Set $\widetilde{I}_1, \ldots, \widetilde{I}_j = \texttt{PartitionAndSample}(G(D), \phi, \hat{\epsilon}, \hat{p})$.
   c. Return $(\widetilde{H}_1, \ldots, \widetilde{H}_k, \widetilde{I}_1, \ldots, \widetilde{I}_j)$.

Lemma 9.1 (`PartitionAndSample`). *Let $G = (V, E)$ be a graph. Let $\widetilde{G}_1, \ldots, \widetilde{G}_k$ be the output of* `PartitionAndSample`$(G, \phi, \hat{\epsilon}, \hat{p})$. *Let $V_1, \ldots, V_k$ be the vertex sets of $\widetilde{G}_1, \ldots, \widetilde{G}_k$, respectively, and let G_0 be the graph with vertex set V and edge set $\partial(V_1, \ldots, V_k)$.*

Then,

(PS.1) $|\partial(V_1, \ldots, V_k)| \leq |E|/2$.

With probability at least $1 - 3n\hat{p}$,

(PS.2) the graph

$$G_0 + \sum_{i=1}^{k} \widetilde{G}_i$$

is a $(1 + \hat{\epsilon})^{1+\log_{29/28} \mathrm{Vol}(V)}$ approximation of G, and

(PS.3) the total number of edges in $\widetilde{G}_1, \ldots, \widetilde{G}_k$ is at most $c_3\epsilon^{-2}|V|\log^{30}(n/p)$, for some absolute constant c_3.

Proof. We first observe that whenever the algorithm calls itself recursively, the volume of the graph in the recursive call is at most 28/29 of the volume of the input graph. So, the recursion depth of the algorithm is at most $\log_{29/28} \mathrm{Vol}(V)$. Property $(PS.1)$ is a consequence of part $(A.2)$ of Theorem 8.1 and this bound on the recursion depth.

We will assume for the rest of the analysis that

1. for every call to `Sample` in line 2, $\widetilde{G}_1$ is a $(1 + \hat{\epsilon})$ approximation of G and the number of edges in $\widetilde{G}_1$ satisfies (S.2),
2. for every call to `Sample` in line 3a, $\widetilde{G}_1 + G(D) + \partial(D, V - D)$ is a $(1 + \hat{\epsilon})$ approximation of G and the number of edges in $\widetilde{G}_1$ satisfies (S.2), and
3. For every call to `ApproxCut` in line 1 for which the set D returned satisfies $\mathrm{Vol}(D) \leq (1/29)\mathrm{Vol}(V)$, there exists a set W containing $V - D$ for which $\Phi_W^G \geq f_2(\phi)$, where f_2 was defined in (14).

First observe that at most n calls are made to `Sample` and `ApproxCut` during the course of the algorithm. By Theorem 8.1, the probability that assumption 3 fails is at most $n\hat{p}$. If assumption 3 never fails, we may apply Theorem 6.1 to prove that assumptions 1 and 2 probably hold, as follows. Consider a subgraph $G(V - D)$ on which `Sample` is called, using $D = \emptyset$ if `Sample` is called on line 2. Assumption 3 tells us that there is a set $W \supseteq V - D$ for which $\Phi_W^G \geq f_2(\phi)$. Theorem 4.1 tells us that the smallest non-zero normalized Laplacian eigenvalue of $G(W)$ is at least λ, where λ is set in line 0. Treating $G(W)$ as the input graph, and $S = V - D$, we may apply Theorem 6.1 to show that assumptions 1 and 2 fail with probability at most $\hat{p}$ each. Thus, all three assumptions hold with probability at least $1 - 3n\hat{p}$.

Property $(PS.3)$, and the existence of the constant c_3, is a consequence of assumptions 1 and 2. Using these assumptions, we will now establish $(PS.2)$ by induction on the depth of the recursion. For a graph G on which `PartitionAndSample` is called, let d be the maximum depth of recursive calls of the algorithm on G, let $\widetilde{G}_1, \ldots, \widetilde{G}_k$ be output of `PartitionAndSample` on

G, and let $V_1, \ldots, V_k$ be the vertex sets of $\widetilde{G}_1, \ldots, \widetilde{G}_k$, respectively. We will prove by induction on d that

$$\sum_{i=1}^{k} \widetilde{G}_i + \partial(V_1, \ldots, V_k) \text{ is a } (1+\hat{\epsilon})^{d+1}\text{-approximation of } G. \tag{19}$$

We base our induction on the case in which the algorithm does not call itself, in which case it returns the output of `Sample` in line 2, and the assertion follows from assumption 1.

Let D be the set of vertices returned by `ApproxCut`. If $D \neq \emptyset$, then $d \geq 1$. We first consider the case in which $\mathrm{Vol}(D) \leq (1/29)\mathrm{Vol}(V)$. In this case, let $H = G(D)$, let $\widetilde{H}_1, \ldots, \widetilde{H}_k$ be the graphs returned by the recursive call to `PartitionAndSample` on H, and let $W_1, \ldots, W_k$ be the vertex sets of $\widetilde{H}_1, \ldots, \widetilde{H}_k$. Let H_0 be the graph on vertex set D with edges $\partial(W_1, \ldots, W_k)$. We may assume by way of induction that

$$H_0 + \sum_{i=1}^{k} \widetilde{H}_i$$

is a $(1+\hat{\epsilon})^d$-approximation of H. We then have

$$\begin{aligned}
G &= G(V-D) + H + \partial(V-D, D) \\
&\preccurlyeq (1+\hat{\epsilon})\left(\widetilde{G}_1 + H + \partial(V-D, D)\right), && \text{by assumption 2,} \\
&\preccurlyeq (1+\hat{\epsilon})\left(\widetilde{G}_1 + (1+\hat{\epsilon})^d\left(\sum_{i=1}^{k}\widetilde{H}_i + H_0\right) + \partial(V-D, D)\right), && \text{by induction,} \\
&\preccurlyeq (1+\hat{\epsilon})^{d+1}\left(\widetilde{G}_1 + \sum_{i=1}^{k}\widetilde{H}_i + H_0 + \partial(V-D, D)\right) \\
&= (1+\hat{\epsilon})^{d+1}\left(\widetilde{G}_1 + \sum_{i=1}^{k}\widetilde{H}_i + \partial(V-D, W_1, \ldots, W_k)\right).
\end{aligned}$$

One may similarly prove

$$(1+\hat{\epsilon})^{d+1} G \succcurlyeq \left(\widetilde{G}_1 + \sum_{i=1}^{k}\widetilde{H}_i + \partial(V-D, W_1, \ldots, W_k)\right),$$

establishing (19) for G.

We now consider the case in which $\mathrm{Vol}(D) > (1/29)\mathrm{Vol}(V)$. In this case, let $H = G(D)$ and $I = G(V-D)$. Let $W_1, \ldots, W_k$ be the vertex sets of $\widetilde{H}_1, \ldots, \widetilde{H}_k$ and let $U_1, \ldots, U_j$ be the vertex sets of $\widetilde{I}_1, \ldots \widetilde{I}_j$. By our inductive hypothesis, we may assume that $\partial(W_1, \ldots, W_j) + \sum_{i=1}^{k}\widetilde{H}_i$ is a $(1+\hat{\epsilon})^d$-approximation of H and that $\partial(U_1, \ldots, U_j) + \sum_{i=1}^{j}\widetilde{I}_i$ is a $(1+\hat{\epsilon})^d$-approximation of I. These two assumptions immediately imply that

$$\partial(W_1, \ldots, W_j, U_1, \ldots, U_j) + \sum_{i=1}^{k}\widetilde{H}_i + \sum_{i=1}^{j}\widetilde{I}_i$$

is a $(1+\hat{\epsilon})^d$-approximation of G, establishing (19) in the second case.

As the recursion depth of this algorithm is bounded by $\log_{29/28}\mathrm{Vol}(V)$, we have established property $(PS.2)$. $\square$

Lemma 9.2 (`UnwtedSparsify`). *For $\epsilon, p \in (0, 1/2)$ and an unweighted graph G with n vertices, let $\widetilde{G}$ be the output of* `UnwtedSparsify`(G, ϵ, p)*. Then,*

(U.1) The edges of $\widetilde{G}$ are a subset of the edges of G; and

with probability at least $1 - p$,

(U.2) $\widetilde{G}$ is a $(1+\epsilon)$-approximation of G, and

(U.3) $\widetilde{G}$ has at most $c_4 \epsilon^{-2} n \log^{31}(n/p)$ edges, for some constant c_4.

Moreover, the expected running time of `UnwtedSparsify` is $O\left(m \log(1/p) \log^{15} n\right)$.

Proof. From $(PS.1)$, we know that the depth of the recursion of `UnwtedSparsify` on G is at most $\log_2 \mathrm{Vol}(V) \le 2 \log n$. So, with probability at least

$$1 - (2 \log n) \cdot 3n\hat{p} = 1 - p,$$

properties $(PS.2)$ and $(PS.3)$ hold for the output of `PartitionAndSample` every time it is called by `UnwtedSparsify`. For the rest of the proof, we assume that this is the case.

Claim $(U.3)$ follows immediately from $(PS.3)$ and the bound on the recursion depth of `UnwtedSparsify`. We prove claim $(U.2)$ by induction on the recursion depth. In particular, we prove that if `UnwtedSparsify` makes d recursive calls to itself on graph G, then the graph $\widetilde{G}$ returned is a $(1 + \epsilon \ln 2/(2 \log n + 1))^d$ approximation of G. We base the induction in the case where `UnwtedSparsify` makes no recursive calls to itself, in which case it returns at line 1 with a 1-approximation.

For $d > 0$, we assume for induction that $\widetilde{G}_0$ is a $(1 + \epsilon \ln 2/2 \log n)^{d-1}$-approximation of G_0. By the assumption that $(PS.2)$ holds, we know that $G_0 + \sum_{i=1}^{k} \widetilde{G}_i$ is a

$$(1 + \hat{\epsilon})^{(1 + \log_{29/28} n^2)} \le (1 + \epsilon \ln 2/(2 \log n))$$

approximation of G, as $\epsilon \ln 2/(2 \log n) \le 1$ (here, we apply the inequality $(1 + x \ln 2/k)^k \le 1 + x$). By following the arithmetic in the proof of Lemma 9.1, we may prove that $\widetilde{G}_0 + \sum_{i=1}^{k} \widetilde{G}_i$ is a $(1 + \epsilon \ln 2/(2 \log n))^d$ approximation of G.

To finish, we observe that

$$(1 + \epsilon \ln 2/(2 \log n))^{2 \log n} \le 1 + \epsilon,$$

for $\epsilon < 1$.

Claim $(U.1)$ follows from the observation that the set of edges of the graph output by `Sample` is a subset of the set of edges of its input.

To bound the expected running time of `UnwtedSparsify`, observe that the bound on the recursion depth of `PartitionAndSample` implies that its expected running time is at most $O(\log n)$ times the expected running time of `ApproxCut` with $\phi = \Omega(1/\log n)$, plus the time required to make the calls to sample, which is at most $O(m)$.

Another multiplicative factor of $O(\log n)$ comes from the logarithmic number of times that `UnwtedSparsify` can call itself during the recursion. ∎

10 Sparsifying Weighted Graphs

In this section, we show how to sparsify graphs whose edges have arbitrary weights. We begin by showing how to sparsify weighted graphs whose edge weights are integers in the range $\{1, \ldots, U\}$. One may also think of this as sparsifying a multigraph. This first result will follow simply from the algorithm for sparsifying unweighted graphs, at a cost of a $O(\log U)$ factor in the number of edges in the sparsifier.

We then explain the obstacle to sparsifying arbitrarily weighted graphs and how we overcome it. We end the section by proving that it is possible to modify our construction of sparsifiers so that for every node the total blow-up in weight of the edges attached to it is bounded.

10.1 Bounded Weights

We recall that we treat an unweighted graph as a graph in which every edge has weight 1, and for clarity we often refer to such a graph as a *weight-1 graph*. Our algorithm for sparsifying graphs with weights in $\{1, \ldots, U-1\}$ works by constructing $\log_2 U$ weight-1 graphs G_i and then expressing G as a sum of $2^i G_i$. Each edge of G appears in the graphs G_i for which the ith bit of the binary expansion of the weight of the edge is 1. We sparsify the graphs G_i independently, and then sum the results.

$\widetilde{G} = \texttt{BoundedSparsify}(G, \epsilon, p)$, $G = (V, E, w)$ has integral weights in $[1, 2^u)$.

1. Decompose G as
$$G = \sum_{i=0}^{u-1} 2^i G_i,$$
where each G_i is a weight-1 graph.
2. For each i, set $\widetilde{G}_i = \texttt{UnwtedSparsify}(G_i, \epsilon, p/u)$.
3. Return $\widetilde{G} = \sum_i 2^i \widetilde{G}_i$.

Lemma 10.1 (`BoundedSparsify`). *For $\epsilon, p \in (0, 1/2)$ and a graph G with integral weights and with n vertices, let $\widetilde{G}$ be the output of* `BoundedSparsify`(G, ϵ, p). *Let $U - 1$ be the maximum weight of an edge in G. Then,*

(B.1) The edges of $\widetilde{G}$ are a subset of the edges of G; and,

with probability at least $1 - p$,

(B.2) $\widetilde{G}$ is a $(1 + \epsilon)$-approximation of G, and

(B.3) $\widetilde{G}$ has at most $c_4 \epsilon^{-2} n \log U \log^{31}(n/p)$ edges.

Moreover, the expected running time of `BoundedSparsify` is $O\left(m \log U \log(1/p) \log^{15} n\right)$.

Proof. Immediate from Lemma 9.2. ∎

10.2 Coping with Arbitrary Weights: Graph Contraction

When faced with an arbitrary weighted graph, we will first approximate the weight of every edge by the sum of a few powers of two. However, if the weights are arbitrary many different powers of two could be required, and we could not construct a sparsifier by treating each power of two separately as we did in `BoundedSparsify`. To get around this problem, we observe that when we are considering edges of a given weight, we can assume that all edges of much greater weight have been contracted. We formalize this idea in Lemma 10.2.

By exploiting this idea, we are able to sparsify arbitrary weighted graphs with at most a $O(\log(1/\epsilon))$-factor more edges than employed in `BoundedSparsify` when $U = n$. Our technique is inspired by how Benczur and Karger [BK96] built cut sparsifiers for weighted graphs out of cut sparsifiers for unweighted graphs.

Given a weighted graph $G = (V, E, w)$ and a partition $V_1, \ldots, V_k$ of V, we define the *map* of the partition to be the function

$$\pi : V \to \{1, \ldots, k\}$$

for which $\pi(u) = i$ if $u \in V_i$. We define the *contraction* of G under π to be the weighted graph $H = (\{1, \ldots, k\}, F, z)$, where F consists of edges of the form $(\pi(u), \pi(v))$ for $(u, v) \in E$, and where the weight of edge $(i, j) \in F$ is

$$z(i, j) = \sum_{(u,v):\pi(u)=i,\pi(v)=j} w(u, v).$$

We do not include self-loops in the contraction, so edges $(u, v) \in E$ for which $\pi(u) = \pi(v)$ do not appear in the contraction.

Given a weighted graph $\widetilde{H} = (\{1, \ldots, k\}, \widetilde{F}, \tilde{z})$, we say that $\widetilde{G} = (V, \widetilde{E}, \tilde{w})$ is a *pullback* of $\widetilde{H}$ under π if

1. $\widetilde{H}$ is the contraction of $\widetilde{G}$ under π, and
2. for every edge $(i, j) \in \widetilde{F}$, $\widetilde{E}$ contains exactly one edge (u, v) for which $\pi(u) = i$ and $\pi(v) = j$.

In the following lemma, we consider a graph in which each of the vertex sets $V_1, \ldots, V_k$ are connected by edges of high weight while all the edges that go between these sets have low weight. We show that one can sparsify the low-weight edges by taking a pullback of an approximation of the contraction of the graph.

Lemma 10.2 (Pullback). *Let $G = (V, E, w)$ be a weighted graph, let $V_1, \ldots, V_k$ be a partition of V, and let π be the map of the partition. Set $E_0 = \partial(V_1, \ldots, V_k)$, $G_0 = (V, E_0, w)$, $E_1 = E - E_0$, and $G_1 = (V, E_1, w)$. For some $\epsilon < 1/2$ let $\widetilde{G}_0$ be a pullback under π of a $(1+\epsilon)$-approximation of the contraction of G_0 under π. Assuming that $c \geq 3$,*

1. *each set of vertices V_i is connected by edges in E_1,*
2. *every edge in E_1 has weight at least $c^2 n^3$, and*
3. *every edge in E_0 has weight 1.*

Then, $\widetilde{G}_0 + G_1$ is an α-approximation of G, for

$$\alpha = (1+\epsilon)(1+1/c)^2.$$

Our proof of Lemma 10.2 uses the following lemma bounding how well a path preconditions an edge. It is an example of a Poincaré inequality [DS91], and it may be derived from the Rank-One Support Lemma of [BH03], the Congestion-Dilation Lemma of [BGH+06], or the Path Lemma of [ST08b]. We include a proof for convenience.

Lemma 10.3. *Let (u,v) be an edge of weight 1, and let F consist of a path from u to v in which the edges on the path have weights $w_1, \ldots, w_k$. Then,*

$$(u,v) \preccurlyeq (1/w_1 + \cdots + 1/w_k)\, F.$$

Proof. Name the vertices on the path 0 through k with vertex 0 replacing u and vertex k replacing v. Let w_i denote the weight of edge $(i, i-1)$. We need to prove that for every vector x,

$$(x(k) - x(0))^2 \leq \left(\sum_{i=1}^{k} \frac{1}{w_i}\right) \sum_{i=1}^{k} w_i (x(i) - x(i-1))^2.$$

For $1 \leq i \leq k$ set $y(i) = \sqrt{w_i}(x_i - x_{i-1})$. The Cauchy-Schwarz inequality now tells us that

$$(x(k) - x(0))^2 = \left(\sum_{i=1}^{k} \sqrt{w_i}(x_i - x_{i-1})/\sqrt{w_i}\right)^2 \leq \left(\sum_{i=1}^{k} (1/\sqrt{w_i})^2\right) \left(\sum_{i=1}^{k} \left(\sqrt{w_i}(x_i - x_{i-1})\right)^2\right),$$

as required. □

Proof of Lemma 10.2. Let H be the contraction of G_0 under π, and let $\widetilde{H}$ be the $(1+\epsilon)$-approximation of H for which $\widetilde{G}_0$ is a pullback.

We begin the proof by choosing an arbitrary vertex v_i in each set V_i. Now, let F be the weighted graph on vertex set $\{v_1, \ldots, v_k\}$ isomorphic to H under the map $i \mapsto v_i$, and let $\widetilde{F}$ be the analogous graph for $\widetilde{H}$. Our analysis will go through an examination of the graphs

$$I \stackrel{\text{def}}{=} F + G_1 \quad \text{and} \quad \widetilde{I} \stackrel{\text{def}}{=} \widetilde{F} + G_1.$$

The lemma is a consequence of the following three statements, which we will prove momentarily:

(a) I is a $(1+1/c)$-approximation of G.

(b) $\widetilde{I}$ is a $(1+\epsilon)$-approximation of I.

(c) $\widetilde{I}$ is a $(1+1/c)$-approximation of $\widetilde{G}_0 + G_1$.

To prove claim (a), consider any edge $(a,b) \in E_0$. As $\pi(a) \neq \pi(b)$, the graph $\frac{1}{cn^2} G_1$ contains a path from a to $v_{\pi(a)}$ and a path from b to $v_{\pi(b)}$. The sum of the lengths of these paths is at most n, and each edge on each path has weight at least cn. So, if we let f denote an edge of weight 1 from $\pi(a)$ to $\pi(b)$, then Lemma 10.3 tells us that

$$(a,b) \preccurlyeq (1/1 + n/cn)\left(f + \frac{1}{cn^2} G_1\right) = (1 + 1/c)\left(f + \frac{1}{cn^2} G_1\right), \tag{20}$$

and

$$f \preccurlyeq (1 + 1/c) \left((a, b) + \frac{1}{cn^2} G_1 \right). \tag{21}$$

As there are fewer than $n^2/2$ edges in E_0, we may sum (20) over all of them to establish

$$G_0 \preccurlyeq (1 + 1/c) \left[F + \frac{1}{2c} G_1 \right].$$

So,

$$\begin{aligned} G_0 + G_1 &\preccurlyeq (1 + 1/c) \left[F + \frac{1}{2c} G_1 \right] + G_1 \\ &\preccurlyeq (1 + 1/c) \left[F + G_1 \right], \end{aligned}$$

as $c \geq 1$. The inequality

$$F + G_1 \preccurlyeq (1 + 1/c) \left[G_0 + G_1 \right],$$

and thus part (a), may be established by similarly summing over inequality (21).

Part (b) is immediate from the facts that $\widetilde{F}$ is a $(1+\epsilon)$-approximation of F, that $I = F + G_1$ and $\widetilde{I} = \widetilde{F} + G_1$.

Part (c) is very similar to part (a). We first note that the sum of the weights of edges in $\widetilde{F}$ is at most $(1 + \epsilon)$ times the sum of the weights of edges in F, and so is at most $(1 + \epsilon)n^2/2$. Now, for each edge (a, b) in $\widetilde{G}_0$ of weight w, there is a corresponding edge $(v_{\pi(a)}, v_{\pi(b)})$ of weight w in $\widetilde{F}$. Let e denote the edge (a, b) of weight w and let f denote the edge $(v_{\pi(a)}, v_{\pi(b)})$ of weight w. As in the proof of part (a), we have

$$e \preccurlyeq (1 + 1/c) \left(f + \frac{w}{cn^2} G_1 \right),$$

and

$$f \preccurlyeq (1 + 1/c) \left(e + \frac{w}{cn^2} G_1 \right).$$

Summing these inequalities over all edges in $\widetilde{E}_0$, adding G_1 to each side, and recalling $\epsilon \leq 1/2$ and $c \geq 3$, we establish part (c). □

We now state the algorithm `Sparsify`. For simplicity of exposition, we assume that the weights of edges in its input are all at most 1. However, this is not a restriction as one can scale down the weights of any graph to satisfy this requirement, apply `Sparsify`, and then scale back up.

The algorithm `Sparsify` first replaces each weight w_e with its truncation to its few most significant bits, z_e. The resulting modified graph is called $\widehat{G}$. As z_e is very close to w_e, little is lost by this substitution. As in `BoundedSparsify`, $\widehat{G}$ is represented as a sum of graphs $2^{-i}G^i$ where each G^i is a weight-1 graph. Because the weight of every edge in $\widehat{G}$ only has a few bits, each edge only appears in a few of the graphs G^i.

Our first instinct would be to sparsify each of the graphs G^i individually. However, this could result in too many edges as sparsifying produces a graph whose number of edges is proportional to its number of vertices, and the sum over i of the number of vertices in each G^i could be large. To get around this problem, we contract all edges of much higher weight before sparsifying. In particular, the algorithm `Sparsify` partitions the vertices into components that are connected by edges of much higher weight. It then replaces each G^i with a pullback of a sparsifier of the contraction of G^i under this partition. In Lemma 10.4 we prove that the sum over i of the number of vertices in the contraction of each G^i will only be a small multiple of n.

$\widetilde{G} = \texttt{Sparsify}(G, \epsilon, p)$, where $G = (V, E, w)$ and $w(e) \leq 1$ for all $e \in E$.

0. Set $Q = \lceil 6/\epsilon \rceil$, $b = 6/\epsilon$, $c = 6/\epsilon$, $\hat{\epsilon} = \epsilon/6$, and $l = \lceil \log_2 2bc^2n^3 \rceil$.
1. For each edge $e \in E$,
    a. choose r_e so that $Q \leq 2^{r_e} w_e < 2Q$,
    b. let q_e be the largest integer such that $q_e 2^{-r_e} \leq w_e$, (and note $Q \leq q_e < 2Q$)
    c. set $z_e = q_e 2^{-r_e}$.
2. Let $\widehat{G} = (V, E, z)$, and express
$$\widehat{G} = \sum_{i \geq 0} 2^{-i} G^i,$$
   where in each graph G^i all edges have weight 1, and each edge appears in at most $\lceil \log_2 2Q \rceil$ of these graphs.
3. Let E^i be the edge set of G^i. Let $E^{\leq i} = \cup_{j \leq i} E^j$. For each i, let $D_1^{\leq i}, \ldots, D_{\eta_i}^{\leq i}$ be the connected components of V under $E^{\leq i}$. For $i = 0$, set $\eta_i = 0$.
4. For each i for which E^i is non-empty,
    a. Let V^i be the set of vertices attached to edges in E^i.
    b. Let $C_1^i, \ldots, C_{k_i}^i$ be the sets of form $D_j^{\leq i-l} \cap V^i$ that are non-empty and have an edge of E^i on their boundary, (that is, the interesting components of V^i after contracting edges in $E^{\leq i-l}$). Let $W^i = \cup_j C_j^i$.
    c. Let π be the map of partition $C_1^i, \ldots, C_{k_i}^i$, and let H^i be the contraction of (W^i, E^i) under π.
    d. $\widetilde{H}^i = \texttt{BoundedSparsify}(H^i, \hat{\epsilon}, p/(2nl))$.
    e. Let $\widetilde{G}^i$ be a pullback of $\widetilde{H}^i$ under π whose edges are a subset of E^i.
5. Return $\widetilde{G} = \sum_i 2^{-i} \widetilde{G}^i$.

Lemma 10.4. *Let k_i denote the number of clusters described by* `Sparsify` *at step 4b. Then,*
$$\sum_i k_i \leq 2nl.$$

Proof. Let η_i denote the number of connected components in the graph $(V, E^{\leq i})$. Each cluster C_j^i has at least one edge of E^i leaving it. As each pair of components under $E^{\leq i-l}$ that are joined by an edge of E^i appear in the same component under $E^{\leq i}$,
$$\eta_i \leq \eta_{i-l} - k_i/2.$$

As the number of clusters never goes negative and is initially at most n, we may conclude

$$\sum_i k_i \leq 2nl.$$

$\square$

Theorem 10.5 (`Sparsify`). *For $\epsilon \in (1/n, 1/3)$, $p \in (0, 1/2)$ and a weighted graph G and with n vertices in which every edge has weight at most 1. Let $\widetilde{G}$ be the output of* `Sparsify`(G, ϵ, p).

(X.1) The edges of $\widetilde{G}$ are a subset of the edges of G; and

with probability at least $1 - p$,

(X.2) $\widetilde{G}$ is a $(1 + \epsilon)$-approximation of G, and

(X.3) $\widetilde{G}$ has at most $c_5 \epsilon^{-2} n \log^{33}(n/p)$ edges, for some constant c_5.

Moreover, the expected running time of `Sparsify` *is $O\left(m \log(1/p) \log^{17} n\right)$.*

Proof. To establish property $(X.1)$, it suffices to show that step 4e can actually be implemented. That is, we need to know that all edges in $\widetilde{H}^i$ can be pulled back to edges of E^i. This follows from $(B.1)$ and the fact that H^i is a contraction of E^i.

We now establish that the graph $\widehat{G}$ is a $(1 + 1/Q)$-approximation of G. We will then spend the rest of the proof establishing that $\widetilde{G}$ approximates $\widehat{G}$. As the weight of every edge in $\widehat{G}$ is less than the corresponding weight in G, we have $\widehat{G} \preccurlyeq G$. On the other hand, for every edge $e \in E$, $w_e \leq (1 + 1/Q) z_e$, so $G \preccurlyeq (1 + 1/Q)\widehat{G}$, and $\widehat{G}$ is a $(1 + 1/Q)$-approximation of G.

From Lemma 10.4, we know that there are at most nl values of i for which $k_i \geq 2$, and so `BoundedSparsify` is called at most nl times. Thus, with probability at least $1 - p$, the output returned by every call to `BoundedSparsify` satisfies properties $(B.2)$ and $(B.3)$, and accordingly we will assume that these properties are satisfied for the rest of the proof.

As each edge set E^i has at most n^2 edges, the weight of every edge in graph H^i is an integer between 1 and n^2. So, by property $(B.3)$, the number of edges in $\widetilde{H}_i$, and therefore in $\widetilde{G}_i$, is at most

$$c_4 \hat{\epsilon}^{-2} k_i \log n^2 \log^{31}(k_i/(p/(2nl))) \leq c_4 \hat{\epsilon}^{-2} k_i \log^{32}(n^2 l/p).$$

Applying Lemma 10.4, we may prove that the number of edges in $\widetilde{G}$ is at most

$$\sum_i c_4 \hat{\epsilon}^{-2} k_i \log^{32}(n^2 l/p) \leq c_4 \hat{\epsilon}^{-2} (2nl) \log^{32}(n^2 l/p) \leq c_5 \epsilon^{-2} n \log^{33}(n/p), \quad \text{as } \epsilon > 1/n,$$

for some constant c_5, thereby establishing $(X.3)$.

To establish $(X.2)$, define for *every* i the weight-1 graph $F^i = (V, E^{\leq i})$, and observe that

$$\sum_{i \geq 0} 2^{-i} F^i = 2\widehat{G}.$$

We may apply $(B.2)$ and Lemma 10.2 to show that

$$\widetilde{G}^i + c^2 n^3 F^{i-l}$$

is a $(1+\hat{\epsilon})(1+1/c)^2$-approximation of $G^i + c^2 n^3 F^{i-l}$. Summing over i while multiplying the ith term by 2^{-i}, we conclude that

$$\sum_{i \geq 0} 2^{-i} \left(\widetilde{G}^i + c^2 n^3 F^{i-l} \right) = \widetilde{G} + c^2 n^3 \sum_{i \geq 0} 2^{-i} F^{i-l} = \widetilde{G} + 2c^2 n^3 2^{-l} \widehat{G}$$

is a $(1+\hat{\epsilon})(1+1/c)^2$-approximation of

$$\sum_{i \geq 0} 2^{-i} \left(G^i + c^2 n^3 F^{i-l} \right) = \widehat{G} + c^2 n^3 \sum_{i} 2^{-i} F^{i-l} = \widehat{G} + 2c^2 n^3 2^{-l} \widehat{G}.$$

Setting

$$\beta \stackrel{\text{def}}{=} 2c^2 n^3 2^{-l} \leq 1/b,$$

we have proved that $\widetilde{G} + \beta \widehat{G}$ is a $(1+\hat{\epsilon})(1+1/c)^2$-approximation of $(1+\beta)\,\widehat{G}$, and by so Proposition 10.6 below, $\widetilde{G}$ is a

$$(1+\hat{\epsilon})(1+1/c)^2(1+\beta)$$

approximation of $\widehat{G}$. Property $(X.2)$ now follows from the facts that $\widehat{G}$ is a $(1{+}1/Q)$-approximation of G, and

$$(1+\hat{\epsilon})(1+1/c)^2(1+\beta)(1+1/Q) \leq (1+\epsilon/6)^5 \leq (1+\epsilon),$$

for $\epsilon < 1/2$.

To bound the expected running time of `Sparsify`, we observe that the time of the computation is dominated by the calls to `BoundedSparsify` and the time required to actually form the graphs H^i. The sets $D_j^{\leq i}$ may be maintained using union-find [Tar75], and so incur a cost of at most $O(n \log n)$ over the course of the algorithm. Each graph H^i may be formed by determining the component of each of its edges, at a cost of $O(\left|E^i\right| \log n)$. So, the time to form the graphs H^i can be bounded by

$$O(\sum_i \left|E^i\right| \log n) = O(m \lceil \log 2Q \rceil \log n) = O(m \log(1/\epsilon) \log n).$$

This is dominated by our upper bound on the time required in the calls to `BoundedSparsify`, which is

$$O\left(\sum_i \left|E^i\right| \log n \lg(1/p) \log^{15} n \right) = O\left(m \log(1/\epsilon) \log n \lg(1/p) \log^{15} n \right) = O\left(m \log(1/p) \log^{17} n \right).$$

□

Proposition 10.6. *If $\beta, \gamma < 1/2$ and $\widetilde{G} + \beta\widehat{G}$ is a $(1+\gamma)$-approximation of $(1+\beta)\widehat{G}$, then $\widetilde{G}$ is a $(1+\beta)(1+\gamma)$-approximation of $\widehat{G}$.*

Proof. We have

$$\widetilde{G} + \beta\widehat{G} \preccurlyeq (1+\gamma)(1+\beta)\widehat{G},$$

which implies

$$\widetilde{G} \preccurlyeq (1+\gamma)(1+\beta)\widehat{G}.$$

On the other hand,

$$
\begin{aligned}
(1+\beta)\widehat{G} \preccurlyeq (1+\gamma)\left(\widetilde{G}+\beta\widehat{G}\right) \quad & \text{implies} \\
(1-\beta\gamma)\widehat{G} \preccurlyeq (1+\gamma)\widetilde{G}, \quad & \text{which implies} \\
\widehat{G} \preccurlyeq \frac{1+\gamma}{1-\beta\gamma}\widetilde{G} & \\
\preccurlyeq (1+\beta)(1+\gamma)\widetilde{G}, &
\end{aligned}
$$

under the conditions $\beta, \gamma < 1/2$. □

10.3 Bounding Blow-Up

When we approximate a graph $G = (V, E, w)$ by a graph $\widetilde{G} = (V, \widetilde{E}, \tilde{w})$ with $\widetilde{E} \subseteq E$, we define the *blow-up* of an edge $e \in E$ by

$$
\text{blow-up}_{\widetilde{G}}(e) \stackrel{\text{def}}{=} \begin{cases} \frac{\tilde{w}_e}{w_e} & \text{if } e \in \widetilde{E}, \text{ and} \\ 0 & \text{otherwise} \end{cases}
$$

Similarly, we define the blow-up of a vertex v to be

$$
\text{blow-up}_{\widetilde{G}}(v) \stackrel{\text{def}}{=} \frac{1}{d_v} \sum_{(u,v)\in E} \text{blow-up}_{\widetilde{G}}((u,v)).
$$

The algorithm in [ST08b] for solving linear equations requires sparsifiers in which every vertex has bounded blow-up. While the sparsifiers output by `UnwtedSparsify` and `BoundedSparsify` satisfy this condition with high probability, the sparsifiers output by `Sparsify` do not. The reason is that nodes of low degree can become part of clusters C_j^i with many edges of E^i on their boundary. These clusters can become vertices of high degree in the contraction by π, and so can become attached to edges of high blow-up when they are sparsified.

This problem may be solved by making two modifications to `Sparsify`. First, we sub-divide the clusters C_j^i so all the vertices in each cluster have approximately the same degree, and so that the degree of every vertex in H^i is at most four times the degree of the vertices that map to it. Then, we set $\widetilde{G}_i$ to be a *random* pullback of $\widetilde{H}_i$ whose edges are a subset of E. That is, for each edge $(c, d) \in \widetilde{H}_i$ we pull it back to a randomly chosen edge $(a, b) \in E$ for which $\pi(a) = c$ and $\pi(b) = d$. In this way we may guarantee with high probability that no vertex has high blow-up. We now describe the corresponding algorithm `Sparsify2` by just listing the lines that differ from `Sparsify`.

$\widetilde{G} = \texttt{Sparsify2}(G, \epsilon, p)$, where $G = (V, E, w)$ has all edge-weights at most 1.

4a. Let ${}^{\delta}V$ be the set of vertices in V with degrees in $[2^{\delta}, 2^{\delta+1})$. Let V^i be the set of vertices attached to edges in E^i. Let ${}^{\delta}V^i$ be the set of vertices in ${}^{\delta}V \cap V^i$.

4b. For each δ, let ${}^{\delta}C_1^i, \ldots, {}^{\delta}C_{k_i^{\delta}}^i$ be the sets of form $D_j^{\leq i-l} \cap {}^{\delta}V^i$ that are non-empty and have an edge of E^i on their boundary. Let $W^i = \cup_{j,\delta} {}^{\delta}C_j^i$. For each set ${}^{\delta}C_j^i$ that has more than $2^{\delta+2}$ edges of E^i on its boundary, sub-divide the set until each part has between 2^{δ} and $2^{\delta+2}$ edges on its boundary. [We will give a procedure to do the subdivision in the paragraph immediately after this algorithm]. Let ${}^{\delta}C_1^i, \ldots, {}^{\delta}C_{t_i^{\delta}}^i$ be the resulting collection of sets.

4c. Let π be the map of partition of W^i by the sets $\left\{{}^{\delta}C_j^i\right\}_{j,\delta}$, and let H^i be the contraction of (W^i, E^i) under π.

4e. Let $\widetilde{H}^i = \texttt{BoundedSparsify}(H^i, \hat{\epsilon}, p/(c_8 nl \log n))$. Let $\widetilde{G}^i$ be a random pullback of $\widetilde{H}^i$ under π whose edges are a subset of E.

We should establish that it is possible to sub-divide the clusters as claimed in step 4b. To see this, recall that each vertex in a set ${}^{\delta}C_j^i$ has degree at most $2^{\delta+1}$. So, if we greedily pull off vertices one by one to form a new set, each time we move a vertex the boundary of the new set will increase by at most $2^{\delta+1}$ and the boundary of the old set will decrease by at most $2^{\delta+1}$. Thus, at the point when the size of the boundary of the new set first exceeds 2^{δ}, the size of the boundary of the old set must be at least $2^{\delta+2} - 2^{\delta} - 2^{\delta+1} \geq 2^{\delta}$. So, one can perform the subdivision in step 4b by a naive greedy algorithm.

Theorem 10.7 (Sparsify2)**.** *For $\epsilon \in (1/n, 1/3)$, $p \in (0, 1/2)$ and a weighted graph G with n vertices, let $\widetilde{G}$ be the output of* $\texttt{Sparsify2}(G, \epsilon, p)$*. Then,*

(Y.1) the edges of $\widetilde{G}$ are a subset of the edges of G; and,

with probability at least $1 - (4/3)p$,

(Y.2) $\widetilde{G}$ is a $(1+\epsilon)$-approximation of G, and

(Y.3) $\widetilde{G}$ has at most $c_6 \epsilon^{-2} n \log^{34}(n/p)$ edges, for some constant c_6,

(Y.4) every vertex has blow-up at most 2.

Moreover, the expected running time of Sparsify2 *is $O\left(m \log(1/p) \log^{17} n\right)$.*

Proof. To prove $(Y.3)$, we must bound the number of clusters, $\sum_{i,\delta} t_i^{\delta}$, produced in the modified step 4b. From Lemma 10.4, we know that

$$\sum_i k_i^{\delta} \leq 2(l \cdot n). \tag{22}$$

To bound $\sum_i t_i^{\delta}$, let $\partial_{E_i}(W)$ denote the set of edges in E_i leaving a set of vertices W. Let S_i^{δ} be the set of j for which ${}^{\delta}C_j^i$ was created by subdivision, and recall that for all $j \in S_i^{\delta}$,

$$\left|\partial_{E_i}\left({}^{\delta}C_j^i\right)\right| \geq 2^{\delta}.$$

So,

$$\sum_{j \in S_i^\delta} \left| \partial_{E_i} \left({}^{\delta}C_j^i \right) \right| \geq 2^\delta (t_i^\delta - k_i^\delta),$$

and

$$\sum_{i,j \in S_i^\delta} \left| \partial_{E_i} \left({}^{\delta}C_j^i \right) \right| \geq 2^\delta \sum_I (t_i^\delta - k_i^\delta). \tag{23}$$

As vertices in ${}^{\delta}V$ have at most $2^{\delta+1}$ edges and each edge of $\widehat{G}$ only appears in at most $\lceil \log 2Q \rceil$ sets E^i,

$$\sum_{i,j \in S_i^\delta} \left| \partial_{E^i} \left({}^{\delta}C_j^i \right) \right| \leq \lceil \log 2Q \rceil 2^{\delta+1} \left| {}^{\delta}V \right|. \tag{24}$$

Combining (23) with (24) and (22), we get

$$\sum_i t_i^\delta \leq 2 \lceil \log 2Q \rceil \left| {}^{\delta}V \right| + 2ln,$$

and so

$$\sum_{\delta,i} t_i^\delta \leq 2 \lceil \log 2Q \rceil n + 2ln \lceil \log 2n \rceil \leq c_8 nl \log n,$$

for some constant c_8. By now applying the analysis from the proof of Theorem 10.5, we may prove that $(Y.2)$ and $(Y.3)$ hold with probability at least $1-p$. Of course, property $(Y.1)$ always holds.

To prove property $(Y.4)$, we note that the blow-up of a vertex v is the sum of $1/d_v$ times the the blow-up of each of its edges. We prove in Lemma 10.8 that the expectation of this sum is 1, and in Lemma 10.9 that each term is bounded by

$$\beta = \frac{1}{48 \log(3n/p)^2}.$$

If the variables were independent, we could apply Theorem 6.8 to prove it is unlikely that v has blow-up greater than 2.

However, the variables are not independent. The blow-up of edges output by `BoundedSparsify` are independent. But, the choice of a random pullback at line 4e introduces correlations in the blow-up of edges. Fortunately, the blow-up of edges attached to v have a negative association (as may be proved by Proposition 8 and Lemma 9 of Dubhashi and Ranjan [DR98]). Thus, by Proposition 7 of [DR98], we may still apply Theorem 6.8, with $\epsilon = 1$ and $\mu = 1$ to show that the

$$\Pr \left[\text{blow-up}_{\widetilde{G}}(v) > 2 \right] \leq e^{-48 \log(3n/p)^2/3}.$$

Applying a union bound over the vertices v, we see that $(Y.4)$ hold with probability at least $1 - p/3$.

The analysis of the running time of `Sparsify2` is similar to the analysis of `Sparsify`, except for the work required to sub-divide sets in step 4b, which we now analyze. Each time a vertex is removed from a set ${}^{\delta}C_j^i$ during the subdivision, the work required by a reasonable implementation is proportional to the degree of that vertex in graph G^i. So, the work required to perform all the subdivisions over the course of the algorithm is at most

$$O\left(\sum_{\delta,i} 2^{\delta+1}\left|S_i^\delta\right|\right).$$

As

$$\partial_{E_i}\left({}^\delta C_j^i\right) \geq 2^\delta$$

whenever we subdivide ${}^\delta C_j^i$, we have

$$\sum_{j\in S_i^\delta} \partial_{E_i}\left({}^\delta C_j^i\right) \geq 2^\delta\left|S_i^\delta\right|.$$

Now, by (24)

$$\sum_i 2^\delta\left|S_i^\delta\right| \leq \lceil\log 2Q\rceil 2^{\delta+1}\left|{}^\delta V\right| \leq 2\lceil\log 2Q\rceil \mathrm{Vol}\left({}^\delta V\right).$$

Thus,

$$\sum_{\delta,i} 2^{\delta+1}\left|S_i^\delta\right| \leq 4\lceil\log 2Q\rceil \mathrm{Vol}\left({}^\delta V\right) = O(m\log(1/\epsilon)).$$

The stated bound on the expected running time of `Sparsify2` follows. ☐

Lemma 10.8. *Let $\widetilde{G} = (V, \widetilde{E}, \tilde{w})$ be the graph output by* `Sparsify2` *on input $G = (V, E, w)$. Then, for every $e \in E$,*

$$\mathbf{E}\left[\text{blow-up}_{\widetilde{\mathrm{G}}}\left(\mathrm{e}\right)\right] \leq 1. \tag{25}$$

Proof. We first observe that

$$\mathbf{E}\left[\text{blow-up}_{\widetilde{G}}\left(e\right)\right] = 1. \tag{26}$$

holds for the graph $\widetilde{G}$ output by `Sample` as it takes a weight-1 graph as input, selects a probability p_e for each edge, and includes it at weight $1/p_e$ with probability p_e. As `UnwtedSparsify` merely partitions its input into edge-disjoint subgraphs and then applies `Sample` to some of them, (26) holds for the output of `UnwtedSparsify` as well.

To show that (26) holds for the graph output by `BoundedSparsify` for each edge $e \in E$ and for each i set

$$w_e^i = \begin{cases} 1 & \text{if } e \in G^i \\ 0 & \text{otherwise.} \end{cases}$$

We have

$$w_e = \sum_i 2^i w_e^i.$$

For the graph $\widetilde{G}_i$ returned on line 2 of `BoundedSparsify`, let $\widetilde{G}^i = (V, \widetilde{E}^i, \tilde{w}^i)$. We have established that

$$\mathbf{E}\left[\tilde{w}_e^i\right] = w_e^i.$$

So,

$$\mathbf{E}\left[\text{blow-up}_{\widetilde{G}}(e)\right] = \mathbf{E}\left[\frac{\sum_i 2^i \tilde{w}_e^i}{w_e}\right] = \frac{\sum_i 2^i \mathbf{E}\left[\tilde{w}_e^i\right]}{w_e} = \frac{\sum_i 2^i w_e^i}{w_e} = 1,$$

establishing (26) for the output of `BoundedSparsify`.

Applying similar reasoning, we may establish (25) for the output of `Sparsify2` by proving that for each edge e in each weight-1 graph G^i, the expected blow-up of e in $\widetilde{G}^i$ is at most 1. If e is not on the boundary of a set ${}^{\delta}C_j^i$, then e will not appear in $\widetilde{G}^i$ and so its blow-up will be zero. If $e = (u, v)$ is on the boundary, then let w_e denote the number of edges $e' = (u', v')$ for which $\pi(u) = \pi(u')$ and $\pi(v) = \pi(v')$. If we let $H = (Y, F, y)$ and $\widetilde{H} = (Y, \widetilde{F}, \tilde{y})$, then $w_e = y_{(\pi(u),\pi(v))}$.

Now, let f be the edge $(\pi(u), \pi(v))$ in H. We know that $\mathbf{E}\left[\text{blow-up}_{\widetilde{H}}(f)\right] = 1$. If f appears in $\widetilde{H}$, then the probability that edge e is chosen in the random pullback is $1/w_e$. As f has weight w_e, we find

$$\mathbf{E}\left[\text{blow-up}_{\widetilde{G}^i}(e)\right] = \frac{1}{w_e}\left(w_e \mathbf{E}\left[\text{blow-up}_{\widetilde{H}^i}(f)\right]\right) = 1.$$

□

Lemma 10.9. *Let $\widetilde{G} = (V, \widetilde{E}, \tilde{w})$ be the graph output by* `Sparsify2` *on input $G = (V, E, w)$. Then, for every $(u, v) \in E$,*

$$\text{blow-up}_{\widetilde{G}}(u, v) \leq \frac{\min(d_u, d_v)}{48 \log(3n/p)^2}. \tag{27}$$

Proof. As in the proof of the previous lemma, we work our way though the algorithms one-by-one. The graph produced by the algorithm `Sample` has blow-up at most $\min(d_u, d_v)/(16 \log(3/p))^2$ for every edge (u, v). As `UnwtedSparsify` only calls `Sample` on subgraphs of its input graph, a similar guaranteed holds for the output of `UnwtedSparsify`. In fact, as `UnwtedSparsify` calls `Sample` with $\hat{p} < p/n$, every edge output by `UnwtedSparsify` actually has blow-up less than

$$\min(d_u, d_v)/(16 \log(3n/p))^2.$$

As `BoundedSparsify` merely calls `UnwtedSparsify` on a collection of graphs that sum to G, the same bound holds on the blow-up of the graph output by `BoundedSparsify`.

To bound the blow-up of edges in the graph output by `Sparsify2`, note that for every i and every vertex a in a graph H^i, the vertices v of the original graph that map to H^i under π satisfy

$$d_v \geq 4 d_a,$$

where d_v refers to the degree of vertex v in the original graph and d_a is the degree of vertex a in graph H^i. So, the blow-up of every edge $(u, v) \in E^i$ satisfies

$$\text{blow-up}_{\widetilde{G}^i}(u, v) \leq \frac{4 \min(d_u, d_v)}{(16 \log(3n/p))^2} = \frac{\min(d_u, d_v)}{48 \log(3n/p)^2}$$

We now measure the blow-up of edges relative to $\widehat{G}$ instead of G, which can only over-estimate their blow-up. The lemma then follows from

$$\text{blow-up}_{\widetilde{G}}(u, v) = \sum_i \frac{2^{-i} \text{blow-up}_{\widetilde{G}^i}(u, v)}{z_{u,v}} \leq \frac{\min(d_u, d_v)}{48 \log(3n/p)^2} \sum_i \frac{2^{-i}}{z_{u,v}} = \frac{\min(d_u, d_v)}{48 \log(3n/p)^2}.$$

□

11 Final Remarks

Since the initial announcement [ST04] of our results, significant improvements have been made in spectral sparsification. Spielman and Srivastava [SS08] have proved that spectral sparsifiers with $O(n \log n/\epsilon^2)$ edges exist, and may be found in time $\widetilde{\mathcal{O}}\left(m \log(nW/\epsilon)\right)$ where W is the ratio of the largest weight to the smallest weight of an edge in the input graph. Their nearly-linear time algorithm relies upon the solution of a logarithmic number of linear systems in diagonally-dominant matrices. Until recently, the only nearly-linear time algorithm for solving such systems was the algorithm in [ST08b], which relied upon the constructions in this paper. Recently, Koutis, Miller and Peng [KMP10] have developed a faster algorithm that does not rely on the sparsifier construction of the present paper. Their algorithm finds α-approximate solutions to Laplacian linear systems in time $O(m \log^2 n \log \alpha^{-1})$. One may remove the dependence on W in the running time of the algorithm of [SS08] through the procedure described in Section 10 of this paper. Batson, Spielman and Srivastava [BSS09] have shown that sparsifiers with $O(n/\epsilon^2)$ edges exist, and present a polynomial-time algorithm that finds these sparsifiers. It is our hope that sparsifiers with so few edges may also be found in nearly-linear time.

Andersen, Chung and Lang [ACL06] and Andersen and Peres [AP09] have improved upon some of the core algorithms we presented in [ST08a] and in particular have improved upon the algorithm `Partition` upon which we based `ApproxCut`. The algorithm of Andersen and Peres [AP09] is both significantly faster and saves a factor of $\log^2 m$ in the conductance of the set it outputs. In particular, it satisfies guarantee $(P.3)$ with the term $O(\tau^2/\log n)$ in place of our function $f_1(\tau)$.

References

[ACL06] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. *Proceedings of the 47th Annual Symposium on Foundations of Computer Science*, pages 475–486, 2006.

[AM01] Dimitris Achlioptas and Frank McSherry. Fast computation of low rank matrix approximations. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 611–618, 2001.

[AP09] Reid Andersen and Yuval Peres. Finding sparse cuts locally using evolving sets. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 235–244, New York, NY, USA, 2009. ACM.

[Axe85] O. Axelsson. A survey of preconditioned iterative methods for linear systems of algebraic equations. *BIT Numerical Mathematics*, 25(1):165–187, March 1985.

[BGH+06] M. Bern, J. Gilbert, B. Hendrickson, N. Nguyen, and S. Toledo. Support-graph preconditioners. *SIAM J. Matrix Anal. & Appl*, 27(4):930–951, 2006.

[BH03] Erik G. Boman and Bruce Hendrickson. Support theory for preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 25(3):694–717, 2003.

[BK96] András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $O(n^2)$ time. In *Proceedings of The Twenty-Eighth Annual ACM Symposium On The Theory Of Computing (STOC '96)*, pages 47–55, May 1996.

[Bol98] Béla Bollobás. *Modern graph theory.* Springer-Verlag, New York, 1998.

[BSS09] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-Ramanujan sparsifiers. In *Proceedings of the 41st Annual ACM Symposium on Theory of computing*, pages 255–262, 2009.

[Che70] J. Cheeger. A lower bound for smallest eigenvalue of laplacian. In *Problems in Analysis*, pages 195–199, In R.C. Gunning editor,, Princeton University Press, 1970.

[Che89] Paul Chew. There are planar graphs almost as good as the complete graph. *J. Comput. Syst. Sci.*, 39:205–219, 1989.

[Chu97] Fan R. K. Chung. *Spectral Graph Theory.* CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997.

[DR98] Devdatt Dubhashi and Desh Ranjan. Balls and bins: a study in negative dependence. *Random Structures and Algorithms*, 13(2):99–124, 1998.

[DS91] Persi Diaconis and Daniel Stroock. Geometric bounds for eigenvalues of markov chains. *The Annals of Applied Probability*, 1(1):36–61, 1991.

[FK81] Z. Füredi and J. Komlós. The eigenvalues of random symmetric matrices. *Combinatorica*, 1(3):233–241, 1981.

[GR01] Chris Godsil and Gordon Royle. *Algebraic Graph Theory.* Graduate Texts in Mathematics. Springer, 2001.

[KMP10] Ioannis Koutis, Gary L. Miller, and Richard Peng. Approaching optimality for solving sdd systems. March 2010. Available at `http://arxiv.org/abs/1003.2958v1`.

[KVV04] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, 2004.

[LPS88] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[Mar88] G. A. Margulis. Explicit group theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators. *Problems of Information Transmission*, 24(1):39–46, July 1988.

[Moh91] Bojan Mohar. The Laplacian spectrum of graphs. In *Graph Theory, Combinatorics, and Applications*, pages 871–898. Wiley, 1991.

[Rag88] Prabhakar Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *J. Comput. Syst. Sci.*, 37:130–143, 1988.

[SJ89] Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing Markov chains. *Information and Computation*, 82(1):93–133, July 1989.

[SS08] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *Proceedings of the 40th annual ACM Symposium on Theory of Computing*, pages 563–568, 2008.

[ST04] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM Symposium on Theory of Computing*, pages 81–90, 2004. Full version available at `http://arxiv.org/abs/cs.DS/0310051`.

[ST08a] Daniel A. Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. *CoRR*, abs/0809.3232, 2008. Available at `http://arxiv.org/abs/0809.3232`. Submitted to SICOMP.

[ST08b] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *CoRR*, abs/cs/0607105, 2008. Available at `http://www.arxiv.org/abs/cs.NA/0607105`. Submitted to SIMAX.

[Tar75] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22(2):448–501, 1975.

[TB97] L. N. Trefethen and D. Bau. *Numerical Linear Algebra.* SIAM, Philadelphia, PA, 1997.

[Tre05] Lucan Trevisan. Approximation algorithms for unique games. *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 197–205, Oct. 2005.

[Vu07] Van Vu. Spectral norm of random matrices. *Combinatorica*, 27(6):721–736, 2007.