Spectral Sparsification of Graphs∗

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

Shang-Hua Teng ‡
Department of Computer Science
Boston University

September 25, 2008

Abstract

We prove that every graph can be approximated by a sparse (re-weighted) subgraph, called a spectral sparsifier. Our notion of approximation 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 present an algorithm that produces spectral sparsifiers in time $\tilde{O}(m)$, where $m$ is the number of edges in the original graph. This construction is a key component of a nearly-linear time algorithm for solving linear equations in diagonally-dominant matrices (Spielman-Teng 2006).

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 [Che86] was motivated by proximity problems in computational geometry to introduce graph spanners. A spanner is a sparse graph where the shortest-path distance between between every pair of vertices is approximately the same in the original graph as in the sparsifier. Benczur and Karger [BK96], motivated by cut problems, 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. Spectral sparsification is a stronger notion than the cut sparsification of Benczur and Karger. 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. We prove that every weighted graph has a spectral sparsifier with $\tilde{O}(n)$ edges, and show how to compute such a spectral sparsifier in $\tilde{O}(m)$ time.

∗This paper, and its companions [ST08] and [ST06], split and expand on the material that previously appeared in the paper [ST03].
†spielman@cs.yale.edu Supported in part by NSF grants CCR-0634957 and CCR-0707522.
‡steng@cs.bu.edu Supported in part by NSF grants CCR0635102 and ITR CCR-0325630. Part of the writing was done at MSR-NE lab.
The Laplacian matrix of a weighted graph \( G = (V, E, w) \) 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

\[
x^T L_G x = \sum_{(u,v) \in E} w(u,v) (x(u) - x(v))^2.
\]

(1)

For more information on the Laplacian matrix of a graph, we refer the reader to one of [Bol98, Mol91, GR01, Chu97].

We say that \( \tilde{G} \) is a \( \sigma \)-approximation of \( G \) if for all \( x \in \mathbb{R}^V \)

\[
\frac{1}{\sigma} x^T L_G x \leq x^T L_G x \leq \sigma x^T L_G x.
\]

(2)

In contrast, 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.

We prove that for every weighted graph \( G = (V, E, w) \) and every \( \epsilon > 0 \), there is a re-weighted subgraph of \( G \) with \( \tilde{O}(n/\epsilon^2) \) edges that is a \((1+\epsilon)\) approximation of \( G \). Moreover, we show how to find such a subgraph in \( \tilde{O}(m) \) time, where \( n = |V| \) and \( m = |E| \). The constants and powers of logarithms hidden in the \( \tilde{O} \)-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. In recent work, Spielman and Srivastava [SS08] have proved that spectral sparsifiers with \( O(n \log n/\epsilon^2) \) edges exist, and may be found in nearly-linear time. However, their nearly-linear time algorithms relies upon the solution of a logarithmic number of linear systems in diagonally-dominant matrices, which we only know how to do in nearly-linear time by using the sparsification algorithm developed in the present paper. Batson, Spielman and Srivastava [BSS08] 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.

## 2 Related Work

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 [ST08], we develop fast routines for partitioning graphs, which we then use in our algorithms for building sparsifiers. Andersen, Chung and Lang [ACL06] have improved upon some of the core algorithms we presented in [ST08]. In the last paper [ST06], 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.

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

\[
\kappa(G, \tilde{G}) \overset{\text{def}}{=} \left( \max_x \frac{x^T L_G x}{x^T L_{\tilde{G}} x} \right) / \left( \min_x \frac{x^T L_G x}{x^T L_{\tilde{G}} x} \right).
\]
So, if $\tilde{G}$ is a $\sigma$-approximation of $G$ then $\kappa(G, \tilde{G}) \leq \sigma^2$. 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 $\epsilon$ by solving $O(\sigma \log(1/\epsilon))$ linear systems in $\tilde{G}$ 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 provide an algorithm for solving linear systems in $L_G$ to $\epsilon$-accuracy in time $O(n^2 \log(1/\epsilon))$. In our paper on solving linear equations [ST06], we show how to get the time bound down to $\tilde{O}(m \log(1/\epsilon))$, where $m$ is the number of non-zero entries in $G$.

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 an algorithm for partitioning a graph into parts of high conductance without removing too many edges. In Section 7, we prove that such partitions exist, and use them to prove the existence of spectral sparsifiers for all unweighted graphs. Employing a graph cutting procedure from [ST08], we develop in Sections 8 and 9 a nearly-linear time algorithm for sparsifying unweighted graphs. We show how to use this algorithm to sparsify weighted graphs in Section 10.

4 Background and Notation

The notation $\tilde{O}(f(n))$, which appears throughout the introduction, means $O(f(n) \log_c f(n))$, for some constant $c$. By log we always mean the logarithm base 2, and we denote the natural logarithm by $\ln$.

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

$$\frac{1}{\sigma} L_{\tilde{G}} \preceq L_G \preceq \sigma L_{\tilde{G}}.$$  

(3)

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

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 the disjoint union of the graphs. It is immediate that $G \preceq \tilde{G}$ and $H \preceq \tilde{H}$ imply

$$G + H \preceq \tilde{G} + \tilde{H}.$$
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 $T$.

For $S \subseteq V$, we define $\text{Vol}(S) = \sum_{i \in S} d_i$. The conductance of a set of vertices $S$, written $\Phi(S)$, is often defined by

$$\Phi(S) \overset{\text{def}}{=} \frac{|E(S, V - S)|}{\min(\text{Vol}(S), \text{Vol}(V - S))}.$$  

The conductance of $G$ is then given by

$$\Phi_G \overset{\text{def}}{=} \min_{\emptyset \neq S \subseteq V} \Phi(S).$$

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 $L_G$, is defined by

$$L_G = D^{-1/2}L_G D^{-1/2}.$$ 

It is well-known that both $L_G$ and $L_G$ 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 $L_G$, written $\lambda_2(L_G)$, to the conductance of $G$.

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

$$2\Phi_G \geq \lambda_2(L_G) \geq \Phi_G^2/2.$$  

## 5 A few examples

### 5.1 Example 1: Complete Graph

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 $x$ 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. The best of them, called Ramanujan graphs [LPSS88, Mar88], 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 $\tilde{G}$ be a Ramanujan graph in which every edge has been given weight $n/d$, then for every $x$ orthogonal to the all-1s vector,

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

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

### 5.2 Example 2: Joined Complete Graphs

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 $\tilde{G}$ 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 $\tilde{G}_1$ and $\tilde{G}_2$ denote $(n/d)$ times a Ramanujan graph on each vertex set, and let $\tilde{G}_3 = G_3$. Recalling the addition we defined on graphs, we have

$$G = G_1 + G_2 + G_3, \quad \text{and}$$

$$\tilde{G} = \tilde{G}_1 + \tilde{G}_2 + \tilde{G}_3.$$

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

$$\frac{1}{\sigma} \tilde{G}_i \preccurlyeq G_i \preccurlyeq \sigma \tilde{G}_i.$$
As $\tilde{G}_3 = G_3$, we have

$$G = G_1 + G_2 + G_3 \preceq \sigma \tilde{G}_1 + \sigma \tilde{G}_2 + G_3 \preceq \sigma \tilde{G}_1 + \sigma \tilde{G}_2 + \sigma G_3 = \sigma \tilde{G}.$$  

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 and 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 $\tilde{G}$ for which $\tilde{G}$ is not a $\sigma$-approximation of $G$ for any small $\sigma$, 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 $\tilde{G}$ will consist of $n$ complete bipartite graphs, connecting all pairs of vertices $(u, i)$ and $(v, j)$ where $v = u \pm 1 \mod n$ or $\{u, v\} = \{1, n-1\}$. The graph $G$ will be identical to the graph $\tilde{G}$, 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 $\tilde{G}$ only differs by one edge, $\tilde{G}$ is a $(1 + 1/2k)$-approximation of $G$ in the notion considered by Benczur and Karger. To show that $\tilde{G}$ is a poor spectral approximation of $G$, consider the vector $x$ given by

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

One can verify that

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

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

6
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 \( \tilde{G} \) 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 \( \tilde{A} \) the adjacency matrix of the sampled graph \( \tilde{G} \). This procedure guarantees that

\[
E[\tilde{A}] = A.
\]

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

\[
p_{i,j} = \min \left( 1, \frac{\delta}{\min(d_i, d_j)} \right), \quad \text{for all edges } (i,j).
\]

The parameter \( \delta \) 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 \( \delta \), then \( \tilde{G} \) will be a good sparsifier of \( G \) with high probability. The actual theorem that we prove is slightly more complicated, as we prove this in the case where we only apply the sampling on a subgraph of \( G \).

**Theorem 6.1.** Let \( \epsilon, p \in (0, 1/2) \) and let \( G = (V, E) \) be a weighted graph whose smallest non-zero normalized Laplacian eigenvalue is at least \( \lambda \). 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

\[
(S, F) = \text{Sample}((S, F), \epsilon, p, \lambda),
\]

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

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

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

\[
\frac{288 \max (\log_2(3/p), \log_2 n)^2}{(\epsilon \lambda)^2} |S|.
\]
\[
\tilde{G} = \text{Sample}(G, \epsilon, p, \lambda)
\]

1. Set \( k = \max \left( \log_2(3/p), \log_2 n \right) \).
2. Set \( \delta = \left( \frac{12k}{\epsilon^2} \right)^2 \).
3. For every edge \((i, j)\) in \(G\), set \( p_{i,j} = \min \left( 1, \frac{\delta}{\min(d_i, d_j)} \right) \).
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 \(\tilde{G}\).

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

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

1. \(\lambda_2(D^{-1/2}LD^{-1/2}) \geq \lambda\), and
2. \(\|D^{-1/2}(L - \tilde{L})D^{-1/2}\| \leq \epsilon\),

then \(\tilde{G}\) is a \(\sigma\)-approximation of \(G\) for

\[
\sigma = \frac{\lambda}{\lambda - \epsilon}.
\]

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

\[
x^T L x = y^T D^{-1/2}LD^{-1/2} y = z^T \left( D^{-1/2}LD^{-1/2} \right) z \geq \lambda \|z\|^2.
\]

We compute

\[
x^T \tilde{L} x = y^T D^{-1/2}\tilde{L}D^{-1/2} y
= z^T D^{-1/2}\tilde{L}D^{-1/2} z
= z^T D^{-1/2}LD^{-1/2} z + z^T D^{-1/2}(\tilde{L} - L)D^{-1/2} z
= z^T D^{-1/2}LD^{-1/2} z \left( 1 + \frac{z^T D^{-1/2}(\tilde{L} - L)D^{-1/2} z}{z^T D^{-1/2}LD^{-1/2} z} \right)
\geq z^T D^{-1/2}LD^{-1/2} z \left( 1 - \frac{\epsilon \|z\|^2}{\lambda \|z\|^2} \right)
= \left( \frac{\lambda - \epsilon}{\lambda} \right) x^T L x.
\]

\(^1\)Recall that the 2-norm of a symmetric matrix is the largest absolute value of its eigenvalues.
We may similarly show that
\[ x^T \tilde{L} x \leq \left( \frac{\lambda + \epsilon}{\lambda} \right) x^T L x \leq \left( \frac{\lambda}{\lambda - \epsilon} \right) x^T L x. \]
The lemma follows from these inequalities. \( \square \)

Let \( A \) be the adjacency matrix of \( G \) and let \( \tilde{A} \) be the adjacency matrix of \( \tilde{G} \), so for each edge \((i,j)\),
\[ \tilde{A}_{i,j} = \begin{cases} 1/p_{i,j} & \text{with probability } p_{i,j} \text{ and} \\ 0 & \text{with probability } 1 - p_{i,j} \end{cases}. \]

To prove Theorem 6.1 we will observe that
\[ \| D^{-1/2} (L - \tilde{L}) D^{-1/2} \| \leq \| D^{-1/2} (A - \tilde{A}) D^{-1/2} \| + \| D^{-1/2} (D - \tilde{D}) D^{-1/2} \|, \]
where \( \tilde{D} \) is the diagonal matrix of the diagonal entries of \( \tilde{L} \). 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 \( k \),
\[ \Pr \left[ \left\| D^{-1/2} (\tilde{A} - A) D^{-1/2} \right\| \geq \frac{2kn^{1/k}}{\sqrt{\delta}} \right] < 2^{-k}. \]

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
\[ \Delta = D^{-1}(\tilde{A} - A), \]
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 \( D^{-1/2} (\tilde{A} - A) D^{-1/2} \) has the same eigenvalues as \( \Delta \). So, it suffices to bound the absolute values of the eigenvalues of \( \Delta \). Rather than trying to upper bound the eigenvalues of \( \Delta \) directly, we will upper bound a power of \( \Delta \)'s trace. As the trace of a matrix is the sum of its eigenvalues, \( \text{Tr} (\Delta^k) \) is an upper bound on the \( k \)th power of every eigenvalue of \( \Delta \), for every even power \( k \).
Lemma 6.4 implies that, for even $k$,
\[
\frac{nk^k}{\delta^{k/2}} \geq \mathbb{E} \left[ \text{Tr} \left( \Delta^k \right) \right] \geq \mathbb{E} \left[ \lambda_{\text{max}} \left( \Delta^k \right) \right].
\]

Applying Markov’s inequality, we obtain
\[
\Pr \left[ \text{Tr} \left( \Delta^k \right) > 2^k \frac{nk^k}{\delta^{k/2}} \right] \leq 1/2^k.
\]

Recalling that the eigenvalues of $\Delta^k$ are the $k$-th powers of the eigenvalues of $\Delta$, and taking $k$-th roots, we conclude
\[
\Pr \left[ \left\| D^{-1/2}(\tilde{A} - A)D^{-1/2} \right\| > 2^k \frac{n^{1/k}k}{\delta^{1/2}} \right] \leq 1/2^k.
\]

\[\square\]

Lemma 6.4. For even $k$,
\[
\mathbb{E} \left[ \text{Tr} \left( \Delta^k \right) \right] \leq \frac{nk^k}{\delta^{k/2}}.
\]

Proof. Recall that the $(v_0, v_k)$ entry of $\Delta^k$ 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
\[
\mathbb{E} \left[ \left( \Delta^k \right)_{v_0, v_k} \right] = \sum_{v_1, \ldots, v_{k-1}} \mathbb{E} \left[ \prod_{i=1}^{k} \Delta_{v_i-1, v_i} \right]. \quad (5)
\]

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 $\Delta_{v_{i-1}, v_i}$ 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 $v_j$
\[
\mathbb{E} \left[ \Delta_{v_i, v_j} \right] = 0.
\]

As $\Delta_{v_i, v_j}$ is independent of every term in $\Delta$ other than $\Delta_{v_j, v_i}$, we see that the term
\[
\mathbb{E} \left[ \prod_{i=1}^{k} \Delta_{v_i-1, v_i} \right], \quad (6)
\]

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 such walks. 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 $\tau$ denote the map from $[k] - T \rightarrow T$, 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

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

indicating the vertex encountered at time $i$.

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

$5 \mapsto 2 \quad 1 \mapsto b$

$6 \mapsto 3 \quad 2 \mapsto c$

$T = \{1, 2, 3, 4, 8\}$

$\tau : \quad 7 \mapsto 4 \quad \sigma : \quad 3 \mapsto d$

$9 \mapsto 8 \quad 4 \mapsto b$

$10 \mapsto 1 \quad 8 \mapsto e$

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

- if $i \in T$, $v_i = \sigma(i)$,
- if $i \not\in T$, and $v_{i-1} = v_{\tau(i)-1}$, then $v_i = v_{\tau(i)}$, and
- if $i \not\in T$, and $v_{i-1} = v_{\tau(i)}$, then $v_i = v_{\tau(i)-1}$.

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

We have

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

(7)

(11)
is independent of the others, and involves a product of the terms \( \Delta_{v_{s-1},v_s} \) and \( \Delta_{v_s,v_{s-1}} \). In Lemma 6.6 we will prove that

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

which implies

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

As the number of choices for \( \sigma(s) \) that result in a walk on the graph of \( A \), and therefore a valid \( \sigma \), is at most \( d_{v_{s-1}} \), one can show by an inductive argument from the last element of \( T \) to the first that

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

As there are at most at most \( 2^k \) choices for \( T \), and at most \( |T|^{k-|T|} \leq |T|^k \) choices for \( \tau \), we may combine these last two inequalities with (7) to obtain

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

The lemma now follows from

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

\[ \square \]

Claim 6.5.

\[ |\Delta_{i,j}| \leq 1/\delta. \]

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

\[ |\Delta_{i,j}| = \frac{1}{d_i} \leq \frac{1}{\min(d_i, d_j)} \leq 1/\delta. \]

On the other hand, with probability \( p_{i,j} \),

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

As \( \Delta_{i,j} \geq 0 \) in this case, we have established \( |\Delta_{i,j}| \leq 1/\delta. \)

\[ \square \]

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

\[
E \left[ \Delta^k_{r,t} \Delta^l_{t,r} \right] \leq \frac{1}{\delta^{k+l-1}} \frac{1}{d_r}.
\]
Proof. First, if \( p_{i,j} = 1 \), then \( \Delta_{i,j} = 0 \). Second, if \( k+l = 1 \), \( E \left[ \Delta_{r,t}^k \Delta_{t,r}^l \right] = 0 \). So, we may restrict our attention to the case were \( k+l \geq 2 \) and \( p_{i,j} < 1 \), which by (4) implies \( p_{i,j} \geq \delta / \min(d_r, d_t) \).

Claim 6.5 tells us that for \( k \geq 1 \),

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

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

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

\[
E \left[ \Delta_{r,t}^k \Delta_{t,r}^l \right] = \frac{1}{d_r d_t} E \left[ (\tilde{A}_{r,t} - 1)^{k+l} \right] 
\]

using \( k+l = 2 \)

\[
= \frac{1}{d_r d_t} \left( p_{r,t} \left( \frac{1 - p_{r,t}}{p_{r,t}} \right)^2 + (1 - p_{r,t}) \right) 
\]

\[
= \frac{1}{d_r d_t} \left( \frac{1 - p_{r,t}}{p_{r,t}} \right) 
\]

\[
\leq \frac{1}{d_r d_t} \left( \frac{\min(d_r, d_t)}{\delta} \right). 
\]

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

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

\[ \square \]

Lemma 6.7. Let \( G \) be a graph and let \( \tilde{G} \) 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 \( \tilde{D} \) be the diagonal matrix of weighed degrees of \( G \). Then,

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

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

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

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 always lies between \( 0 \) and \( d_i/\delta \), we may apply the variant of the Chernoff bound given in Theorem 6.8 to show that

\[
\Pr \left[ \tilde{d}_i > (1 + \epsilon)d_i \right] \leq e^{-\delta^2/3}. 
\]

The lemma now follows by taking a union bound over \( i \).  \[ \square \]
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 $\alpha_i$ with probability $p_i$ and 0 with probability $1 - p_i$. Let $X = \sum_i X_i$ and $\mu = \mathbb{E}[X] = \sum \alpha_i p_i$. Then,

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

For $\epsilon < 1$, this probability is 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 $\tilde{L}$, $\tilde{A}$ and $\tilde{D}$ be the corresponding matrices for $\tilde{G}$. The matrices $L$ and $\tilde{L}$ 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

$$\left\| D^{-1/2}(L - \tilde{L})D^{-1/2} \right\| = \left\| D(S)^{-1/2}(L(S) - \tilde{L}(S))D(S)^{-1/2} \right\|$$

$$\leq \left\| D(S)^{-1/2}(A(S) - \tilde{A}(S))D(S)^{-1/2} \right\| + \left\| D(S)^{-1/2}(D(S) - \tilde{D}(S))D(S)^{-1/2} \right\|.$$ 

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

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

we find

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

Applying Lemma 6.7 to the second term, we find

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

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

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

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

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

using $\epsilon \leq 1/2$. 

---

14
Finally, we use Theorem 6.8 to bound the number of edges in $\tilde{F}$. 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 $\tilde{F}$. Using $d_i$ to denote the degree of vertex $i$ in $G(S)$, we have

$$E \left[ \sum_{(i,j) \in F} X_{(i,j)} \right] = \delta \sum_{(i,j) \in F} \frac{1}{\min(d_i, d_j)} \leq \delta \sum_{(i,j) \in F} \left( \frac{1}{d_i} + \frac{1}{d_j} \right) = \delta \sum_{i \in S} \sum_{j : (i,j) \in F} \frac{1}{d_i} = \delta |S|.$$ 

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

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

\[ \square \]

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. Finally, the analysis of our algorithm relies on most of the analysis 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 conductance of a set $S$ in the subgraph induced by $B \subseteq V$ by

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

and

$$\Phi^G_B \overset{\text{def}}{=} \min_{S \subseteq B} \Phi^G_B (S).$$

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

\[ ^2 \text{Recall our algorithmic objective is to design a nearly-linear time algorithm for spectral sparsification. In this context, a quadratic-time computation is infeasible.} \]
We introduce the notation $G\{B\}$ to denote graph $G(B)$ to which self-loops have been added so that every vertex in $G\{B\}$ has the same degree as in $G$, so for $S \subseteq B$

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

Because $\Phi^{G\{B\}}$ measures volume by degrees in $G$, if $G(B)$ is the subgraph of $G$ induced on the vertices in $B$, then

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

So, when we prove lower bounds on $\Phi^{G\{B\}}$, 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 **$\phi$-decomposition** if $\Phi^{G\{A_i\}} \geq \phi$ 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:

$$E \cap \bigcup_{i \neq j} (A_i \times A_j).$$

We say that a decomposition $(A_1, \ldots, A_k)$ is a **$\lambda$-spectral decomposition** if the smallest non-zero normalized Laplacian eigenvalue of $A_i$ is at least $\lambda$, for all $i$. By the Cheeger’s inequality (Theorem 4.1), every $\phi$-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 $(6\log_{4/3}2m)^{-1}$-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 $\Omega(\log^6 n)$. It remains to sparsify the edges bridging components. If only $\tilde{O}(n)$ 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 $\Omega(\log^7 n)$.

Recently, Batson, Spielman and Srivastava \[BSS08\] have shown that 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 $\phi$ is small, then the graph induced on the complement
has conductance almost $\phi$. 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 $\text{Vol}(S)$ among those satisfying

(L.1) $\text{Vol}(S) \leq \text{Vol}(B)/2$, and

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

If $\text{Vol}(S) = \alpha \text{Vol}(B)$, with $\alpha \leq 1/3$ 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 (L.1) and (L.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

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

and

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

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

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

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

We begin by observing that

$$|E(R \cup S, B - (R \cup S))| \leq |E(S, B - S)| + |E(R, B - S - R)|$$

$$< \phi \text{Vol}(S) + (\phi\beta) \text{Vol}(R).$$

(9)

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

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

which implies

$$\Phi^G_B(R \cup S) < \phi.$$

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

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

(10)
which implies \( \text{Vol}(B - T) > \text{Vol}(S) \). To prove (10), compute

\[
\text{Vol}(T) = \text{Vol}(S) + \text{Vol}(R) \\
\leq \text{Vol}(S) + (1/2)(\text{Vol}(B) - \text{Vol}(S)) \\
= (1/2)\text{Vol}(B) + (1/2)\text{Vol}(S) \\
= \left(\frac{1 + \alpha}{2}\right)\text{Vol}(B).
\]

To upper bound the conductance of \( T \), we compute

\[
|E(T, B - T)| < \phi \text{Vol}(S) + (\phi \beta) \text{Vol}(R) \quad \text{(by (9))} \\
\leq \phi \text{Vol}(S) + (\phi \beta)(\text{Vol}(B) - \text{Vol}(S))/2 \\
= \phi \text{Vol}(B)(\alpha + \beta(1 - \alpha)/2)
\]

So,

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

by our choice of \( \beta \).

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

\[
\text{idealDecomp}(B)
\]

0. Set \( \phi = \left(2 \log_{4/3} \text{Vol}(V)\right)^{-1} \).

1. If \( \Phi^G_B \geq \phi \), then return \( B \). Otherwise, proceed.

2. Let \( S \) be the subset of \( B \) maximizing \( \text{Vol}(S) \) satisfying (L.1) and (L.2).

3. If \( \text{Vol}(S) \leq \text{Vol}(B)/4 \), return \((B - S, \text{idealDecomp}(S))\).

4. else, return \((\text{idealDecomp}(B - S), \text{idealDecomp}(S))\).

\textbf{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 \text{idealDecomp}(G). 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 \( \log_{4/3} \text{Vol}(V) \) and that at most a \( \phi \) fraction of the edges are added to \( \partial(A_1, \ldots, A_k) \) at each level of the recursion. So, \(|\partial(A_1, \ldots, A_k)| \leq \phi \log_{4/3} \text{Vol}(V) \). The theorem now follows by setting \( \phi = \left(2 \log_{4/3} \text{Vol}(V)\right)^{-1} = (2 \log_{4/3} 2m)^{-1} \).
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 [ST08]. 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 use an algorithm satisfying such a guarantee to certify that the complement of a small cut has high conductance. If we were to just apply the algorithm until it cannot cut any more, as was done by Kannan, Vempala and Vetta [KVV04], our algorithm could take at least quadratic time.

**Theorem 8.1 (ApproxCut).** Let $\phi, p \in (0, 1)$ and let $G$ be a graph. Let $D$ be the output of $\text{ApproxCut}(G, \phi, p)$. Then

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

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

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

(A.3.a) $\text{Vol}(D) \geq \frac{1}{12}\text{Vol}(V)$, or

(A.3.b) there exists a set $W \supseteq V - D$ for which $\Phi_W^G \geq f_2(\phi)$, where

$$f_2(\phi) \overset{\text{def}}{=} \frac{c_2 \phi^3}{\log^2 m},$$

for some absolute constant $c_2$.

Moreover, the expected running time of **ApproxCut** is at most $O(m \log(1/p) \ln^5(\text{Vol}(V))/\phi^4)$.

The code for **ApproxCut** follows. It relies on a routine called **Partition2** which in turn relies on a routine called **Partition** from [ST08].

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

(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 $j \leq r$ and $\text{Vol}(V_j) \geq (4/5)\text{Vol}(V)$,

(a) Set $j = j + 1$.

(b) Set $D_j = \text{Partition2}(G\{V_{j-1}\}, (2/23)\phi, p, 2r, \epsilon)$

(c) Set $V_j = V_{j-1} - D_j$.

(3) Set $D = D_1 \cup \cdots \cup D_j$. 
The algorithm Partition from [ST08], satisfies the following theorem.

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

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

(P.2) If $D \neq \emptyset$ then $\Phi_V(D) \leq \phi$, and

(P.3) For some absolute constant $c_1$ and

$$f_1(\phi) \overset{\text{def}}{=} \frac{c_1 \phi^2}{\log^2 m},$$

for every set $S$ satisfying

$$\text{Vol}(S) \leq \frac{\text{Vol}(V)}{2} \quad \text{and} \quad \Phi_V(S) \leq f_1(\phi),$$

with probability at least $1 - p$ either

(P.3.a) $\text{Vol}(D) \geq \frac{1}{4} \text{Vol}(V)$, or

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

Moreover, the expected running time of Partition is at most $O\left(\frac{m \log(1/p) \ln^5(\text{Vol}(V))/\phi^4}{\epsilon^2}\right)$.

If either (P.3.a) or (P.3.b) occur, we say that Partition succeeds for $S$.

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$(G, \phi, p, \epsilon)$, where $G$ is a graph, $\phi, p \in (0, 1)$ and $\epsilon \in (0, 1/2)$. Then

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

(Q.2) If $D \neq \emptyset$ then $\Phi_V(D) \leq \phi$, and

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

$$\text{Vol}(S) \leq \frac{\text{Vol}(V)}{2} \quad \text{and} \quad \Phi_V(S) \leq \epsilon f_1(\phi)/9,$$

with probability at least $1 - p$, either
\((Q.3.a)\) \(\text{Vol}(D) \geq (1/5)\text{Vol}(V)\), or
\((Q.3.b)\) \(\text{Vol}(S \cap D) \geq (1 - \epsilon)\text{Vol}(S)\).

Moreover, the expected running time of Partition2 is \(O(m \log(1/\epsilon) \log(1/\epsilon)/p) \ln^5(\text{Vol}(V))/\phi^4)\).

If either \((Q.3.a)\) or \((Q.3.b)\) occur, we say that Partition2 succeeds for \(S\).

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

\[\text{Proof.}\] Let \(j^*\) be such that \(D = D_1 \cup \cdots \cup D_{j^*}\). To prove \((Q.3)\), let \(\nu = \text{Vol}((D_1 \cup \cdots \cup D_{j^*-1}))/\text{Vol}(V)\). As \(\text{Vol}(W_{j^*-1}) \geq (4/5)\text{Vol}(V)\), \(\nu \leq 1/5\). By \((P.1)\), \(\text{Vol}(D_{j^*}) \leq (7/8)\text{Vol}(W_{j^*-1})\), so

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

To establish \((Q.2)\), we first compute

\[
|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^*} \left(\phi/9\right) \text{Vol}(D_i, W_{i-1} - D_i) \quad \text{(by \((P.2)\) and line 1b of ApproxCut)}
\leq \sum_{i=1}^{j^*} \left(\phi/9\right) \text{Vol}(D_i)
= \left(\phi/9\right) \text{Vol}(D).
\]

So, if \(\text{Vol}(D) \leq \text{Vol}(V)/2\), then \(\Phi_V(D) \leq \phi/9\). On the other hand, we established above that \(\text{Vol}(D) \leq (9/10)\text{Vol}(V)\), from which it follows that

\[
\text{Vol}(V - D) \geq (1/10)\text{Vol}(V) \geq (1/10)(10/9)\text{Vol}(D) = (1/9)\text{Vol}(D).
\]

So,

\[
\Phi_V(D) = \frac{|E(D, V - D)|}{\min(\text{Vol}(D), \text{Vol}(V - D))} \leq g \frac{|E(D, V - D)|}{\text{Vol}(D)} \leq \phi.
\]

To prove \((Q.3)\), let \(S\) be a set satisfying \((E3)\), and let \(S_j = S \cap W_j\). We need to prove that with probability at least \(1 - p\), either \(\text{Vol}(W_{j^*}) \leq (4/5)\text{Vol}(V)\) or \(\text{Vol}(S_{j^*}) \leq \epsilon\text{Vol}(S)\). If neither of these inequalities hold, then

\[j^* = r, \quad \text{Vol}(W_r) \geq (4/5)\text{Vol}(V), \quad \text{and} \quad \text{Vol}(S_r) > \epsilon\text{Vol}(S)\]

So, there must exist a \(j\) for which \(\text{Vol}(S_{j+1}) \geq (1/2)\text{Vol}(S_j)\). But,

\[
\Phi^G\{W_j\}(S_j) = \Phi_{W_j}(S_j) = \frac{\partial W_j(S_j)}{\min(\text{Vol}(S_j), \text{Vol}(W_j - S_j))} \leq \frac{\partial V(S)}{\min(\text{Vol}(S), \text{Vol}(W_j - S_j))} \leq \frac{\partial V(S)}{\epsilon\text{Vol}(S)} = (1/\epsilon)\Phi_V(S) \leq f_1(\phi),
\]

21
where the second-to-last equality follows from \( \operatorname{Vol}(W_j - S_j) \geq \operatorname{Vol}(V)(4/5 - 1/2) \) and \( \epsilon \operatorname{Vol}(S) \leq \operatorname{Vol}(V)/4 \). So, \( S_j \) satisfies conditions (12), but \( \text{Partition} \) 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 \( \text{Partition2} \) is immediate from the bound on the running time of \( \text{Partition} \).

The rest of this section is devoted to the proof of Theorem 8.1, with all but one line devoted to part (A.3). To prove the existence of the set \( W \), we will inductively construct a sequence of sets \( W_0, \ldots, W_r, U_0, \ldots, U_r \) and \( S_0, \ldots, S_r \), and parameters \( \sigma_0, \ldots, \sigma_r, \theta_0, \ldots, \theta_r \) by the procedure:

1. \( W_0 = V_0 = V \).
2. \( \sigma_0 = (2/27)\epsilon f_1(\phi) \).
3. for \( i = 0 \) to \( r \),
   - If \( W_i \) contains a set \( S_i \) such that \( \operatorname{Vol}(S_i) \leq (1/2)\operatorname{Vol}(W_i) \) and \( \Phi_{W_i}(S_i) \leq \sigma_i \),
     set \( S_i \) to be such a set of maximum size. If there is no such set, stop the procedure.
     and set \( W = W_i \). If \( \operatorname{Vol}(S_i) \geq \operatorname{Vol}(W_i)/3 \), stop the procedure.
   - \( U_i = W_i - S_i \).
   - \( \theta_i = \left(1 - 3\frac{\operatorname{Vol}(S_i)}{\operatorname{Vol}(W_i)}\right)\sigma_i \).
   - \( \sigma_{i+1} = (1 - 2\epsilon)\theta_i \).
   - \( W_{i+1} = U_i \cup V_{i+1} \).
4. Set \( W = W_r \).

If it is ever the case that \( \operatorname{Vol}(S_i) \geq \operatorname{Vol}(W_i)/3 \), then this procedure will fail to produce a set \( W \). But, this will not concern us as we will show that in this case the algorithm \( \text{ApproxCut} \) will probably output a large cut.

**Claim 8.4.** For all \( i \geq 0 \), \( W_{i+1} = U_i \cup (V_{i+1} \cap S_i) \).

**Proof.** We have
\[
V_{i+1} \subseteq V_i \subseteq W_i = U_i \cup S_i.
\]
So,
\[
V_{i+1} = (V_{i+1} \cap U_i) \cup (V_{i+1} \cap S_i),
\]
from which the claim follows. \( \square \)

**Claim 8.5.**
\[
\Phi_{U_i} \geq \theta_i.
\]

**Proof.** Follows immediately from Lemma 7.2 and the definitions of \( S_i \) and \( \theta_i \). \( \square \)
So, the graphs induced on the sets $U_i$ have high conductance. We think of the sets $W_i$ as these high-conductance sets plus the part of $V$ that has not yet been cut out.

**Lemma 8.6.** If

$$\text{Vol}(D) \leq (1/12)\text{Vol}(V)$$

and

1. for all $i$ for which $S_i$ satisfies conditions (13) in $V_i$, the $i$th call to $\text{Partition2}$ succeeds for $S_i$, and

2. for all $i$ for which $V_i - S_i$ satisfies conditions (13) in $V_i$, the $i$th call to $\text{Partition2}$ succeeds for $V_i - S_i$,

then the procedure outputs a set $W$, and

$$\Phi^G_W \geq f_2(\phi).$$

**Proof.** We show that the conditions of the lemma imply

(a) $\text{Vol}(W_i) \geq \text{Vol}(V_i) \geq (5/6)\text{Vol}(V)$,

(b) $\text{Vol}(U_i) \geq (5/12)\text{Vol}(V)$,

(c) $\Phi_{V_i}(S_i) \leq (3/2)\sigma_i$,

(d) $\text{Vol}(S_i) \leq (1/2)\text{Vol}(V_i)$,

(e) $\text{Vol}(V_{i+1} \cap S_i) \leq \epsilon \text{Vol}(S_i)$,

(f) $\text{Vol}(S_i) \leq (5/48)\text{Vol}(V)$,

(g) $\text{Vol}(S_{i+1}) \leq (1/2)\text{Vol}(S_i)$,

(h) $\theta_r \geq (1/4)(1 - 2\epsilon)^r \sigma_0$, if $W = W_r$

Note that condition (f) along with $V_i \subseteq W_i$ implies that the procedure actually does output a set $W$.

To prove (a), observe

$$\text{Vol}(W_i) \geq \text{Vol}(V_i) \geq \text{Vol}(V) - \text{Vol}(D) \geq (5/6)\text{Vol}(V).$$

Part (b) now follows from (a) and $\text{Vol}(U_i) \geq (1/2)\text{Vol}(W_i)$, which follows from the definitions of $S_i$ and $U_i$.

To establish (c), note that $V_i \subseteq W_i$ and the definition of $S_i$ imply

$$|\partial_{V_i}(S_i)| \leq |\partial_{W_i}(S_i)| \leq \sigma_i \min(\text{Vol}(S_i), \text{Vol}(W_i - S_i)) \leq \sigma_i \min(\text{Vol}(S_i), \text{Vol}(V - S_i)).$$

As $\text{Vol}(V_i) \geq (5/6)\text{Vol}(V)$ and $\text{Vol}(S_i) \leq (1/2)\text{Vol}(W_i) \leq (1/2)\text{Vol}(V)$,

$$\text{Vol}(V - S_i) \leq (3/2)\text{Vol}(V_i - S_i).$$
So,

\[ \min (\Vol (S_i), \Vol (V - S_i)) \leq (3/2) \min (\Vol (S_i), \Vol (V_i - S_i)), \]

and

\[ |\partial V_i (S_i)| \leq (3/2) \sigma_i \min (\Vol (S_i), \Vol (V_i - S_i)), \]

which is equivalent to (c).

Assume by way of contradiction that (d) is false. Then \( \Vol (V_i - S_i) \leq (1/2) \Vol (V_i) \) and part (c) implies

\[ \Phi_{V_i} (V_i - S_i) = \Phi_{V_i} (S_i) \leq (3/2) \sigma_i \leq (3/2) \sigma_0 \leq \epsilon f_1 (\phi) / 9. \]

So, \( V_i - S_i \) satisfies conditions (13), and the success of the \( i \)th call to \texttt{Partition2} implies

\[ \Vol (D) \geq \Vol (D_i) \geq \min (\{(1 - \epsilon) (\Vol (V_i) - \Vol (S_i)), (1/5) \Vol (V_i)\}). \] (15)

But,

\[ \Vol (S_i) \leq (1/2) \Vol (W_i) \leq (1/2) \Vol (V) < (1/2)(6/5) \Vol (V_i) = (3/5) \Vol (V_i). \]

So,

\[ (1 - \epsilon) (\Vol (V_i) - \Vol (S_i)) \geq (1/2)(2/5) (\Vol (V_i)) \geq (1/10)(5/6) (\Vol (V)) = (1/12) \Vol (V), \]

which means that (15) contradicts (14).

To prove part (e), note that parts (c) and (d) imply that \( S_i \) satisfies conditions (13), and so (f) follows from the assumption that the call to \texttt{Partition2} for \( S_i \) succeeds and (14).

To prove part (f), note that success of the call to \texttt{Partition2} for \( S_i \) implies

\[ (1 - \epsilon) \Vol (S_i) \leq \Vol (D_i) \leq (1/12) \Vol (V), \]

which establishes (f) for

\[ \epsilon \leq 1/5. \]

The proof of part (g) is the most interesting. Define \( \alpha \) so that \( \Vol (S_{i+1}) = \alpha \Vol (S_i) \). As \( S_{i+1} \subset W_{i+1} = U_i \cup (V_{i+1} \cap S_i) \) and \( \Vol (V_{i+1} \cap S_i) \leq \epsilon \Vol (S_i), \)

\[ \Vol (S_{i+1} \cap U_i) \geq (\alpha - \epsilon) \Vol (S_i). \]

From parts (b) and (f) and Claim 8.5, we have

\[ |\partial U_i (S_{i+1})| \geq \theta_i \min (\Vol (S_{i+1} \cap U_i), \Vol (U_i - (S_{i+1} \cap U_i))) \]

\[ = \theta_i \Vol (S_{i+1} \cap U_i) \geq \theta_i (\alpha - \epsilon) \Vol (S_i). \] (16)

As \( U_i \subseteq W_{i+1}, \)

\[ |\partial U_i (S_{i+1})| \leq |\partial W_{i+1} (S_{i+1})| \leq \sigma_{i+1} \Vol (S_{i+1}) = \sigma_{i+1} \alpha \Vol (S_i). \] (17)

Combining (16) and (17) provides

\[ \theta_i (\alpha - \epsilon) \Vol (S_i) \leq \sigma_{i+1} \alpha \Vol (S_i). \]
As \( \sigma_{i+1} = (1 - 2\epsilon)\theta_i \), this implies \( \alpha \leq 1/2 \), which is equivalent to (f).

To establish (h), first note that

\[
\theta_r = \sigma_0(1 - 2\epsilon)\prod_{i=0}^r \left(1 - \frac{3\text{Vol}(S_i)}{\text{Vol}(W_i)}\right),
\]

Claim (h) now follows from

\[
\prod_{i=0}^r \left(1 - \frac{3\text{Vol}(S_i)}{\text{Vol}(W_i)}\right) \geq 1 - \sum_{i=0}^r \frac{3\text{Vol}(S_i)}{(5/6)\text{Vol}(V)} \\
= 1 - \frac{18}{5} \sum_{i=0}^r \frac{\text{Vol}(S_i)}{\text{Vol}(V)} \\
\geq 1 - \frac{18}{5} \frac{2\text{Vol}(S_0)}{\text{Vol}(V)}, \text{ by (g),} \\
\geq 1 - \frac{36}{5} \frac{5}{48} \text{, by (f),} \\
= \frac{1}{4}.
\]

By our choice of \( \epsilon = \min(1/5, 1/2r) \), we then have

\[
\theta_r \geq \frac{1}{15} \sigma_0 = \frac{1}{16} \frac{2}{27} f_1(\phi) \geq \frac{c_2 \phi^2}{\log^3 m},
\]

for some constant \( c_2 \).

If the procedure stops before the \( r \)th iteration, it is because \( W = W_i \), where \( W_i \) is a set satisfying \( \Phi^G_{W_i}(S_i) > \sigma_i \). By truncating the proof of claim (h), we can prove that

\[
\sigma_i \geq \frac{c_2 \phi^2}{\log^3 m}.
\]

If \( W = W_r \), we use (g) and \( \text{Vol}(S_0) \leq (1/2)\text{Vol}(V) = m \), to show that

\[
\text{Vol}(S_r) = 0,
\]

and

\[
W_r = U_r.
\]

So, \( \Phi^G_{W_r} = \Phi^G_{U_r} \), and the lemma follows from (h) and Claim 8.5.

**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), consider the procedure for producing the set \( W \). For each set \( S_i \) produced by the procedure, the probability that the \( i \)th call to Partition2 fails to succeed for \( S_i \) or \( V_i - S_i \) is at most \( p/2r \), so the probability all these calls succeed is at least \( 1 - p \). If all these calls succeed, then Lemma 8.6 tells us that either

\[
\text{Vol}(D) > (1/12)\text{Vol}(V),
\]

and
in which case (A.3.a) holds, or
\[ \Phi_W \geq f_2(\phi). \]

Let \( i \) be the index for which \( W = W_i. \) As
\[ V - D \subseteq V_i \subseteq W = W, \]
the set \( W \) satisfies (A.3.b).

9 Sparsifying Unweighted Graphs

We now show how to use the algorithms \textsc{ApproxCut} and \textsc{Sample} to sparsify unweighted graphs. More precisely, we treat every edge in an unweighted graph as an edge of weight 1. The algorithm follows the outline described in Section 7.2.

\[ \tilde{G} = \text{UnwtedSparsify}(G, \epsilon, p) \]

1. If \( \text{Vol}(V) \leq c_3 n \log^{22} n \), return \( G \) (where \( c_3 \) is set in the proof of Lemma 9.1).
2. Set \( \phi = \left( 2 \log_{12/11} \text{Vol}(V) \right)^{-1}, \tilde{p} = p/6n \log_2 n, \) and \( \tilde{\epsilon} = \frac{\epsilon (\ln 2)^2}{(1 + 2 \log_{12/11} \log n) (2 \log n)}. \)
3. Set \( (\tilde{G}_1, \ldots, \tilde{G}_k) = \text{PartitionAndSample}(G, \phi, \tilde{\epsilon}, \tilde{p}). \)
4. Let \( V_1, \ldots, V_k \) be the vertex sets of \( \tilde{G}_1, \ldots, \tilde{G}_k, \) respectively, and let \( G_0 \) be the graph with vertex set \( V \) and edge set \( \partial(V_1, \ldots, V_k). \)
5. Set \( \tilde{G}_0 = \text{UnwtedSparsify}(G_0, \epsilon, p). \)
6. Set \( \tilde{G} = \sum_{i=0}^k \tilde{G}_i. \)

\( (\tilde{G}_1, \ldots, \tilde{G}_k) = \text{PartitionAndSample}(G, \phi, \hat{\epsilon}, \hat{p}) \)

0. Set \( \lambda = f_2(\phi)^2/2, \) where \( f_2 \) is defined in (11).
1. Set \( D = \text{ApproxCut}(G, \phi, \hat{p}). \)
2. If \( D = \emptyset, \) return \( \tilde{G}_1 = \text{Sample}(G, \hat{\epsilon}, \hat{p}, \lambda). \)
3. Else, if \( \text{Vol}(D) \leq (1/12) \text{Vol}(V) \)
   a. Set \( \tilde{G}_1 = \text{Sample}(G(V - D), \hat{\epsilon}, \hat{p}, \lambda) \)
   b. Return \( (\tilde{G}_1, \text{PartitionAndSample}(G(D), \phi, \hat{\epsilon}, \hat{p})). \)
4. Else,
   a. Set \( \tilde{H}_1, \ldots, \tilde{H}_k = \text{PartitionAndSample}(G(V - D), \phi, \hat{\epsilon}, \hat{p}). \)
   b. Set \( \tilde{I}_1, \ldots, \tilde{I}_j = \text{PartitionAndSample}(G(D), \phi, \hat{\epsilon}, \hat{p}). \)
   c. Return \( (\tilde{H}_1, \ldots, \tilde{H}_k, \tilde{I}_1, \ldots, \tilde{I}_j). \)
Lemma 9.1 (PartitionAndSample). Let \( \tilde{G}_1, \ldots, \tilde{G}_k \) be the output of PartitionAndSample\((G, \phi, \hat{\epsilon}, \hat{p})\). Let \( V_1, \ldots, V_k \) be the vertex sets of \( \tilde{G}_1, \ldots, \tilde{G}_k \), 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 \( 1 - 3n\hat{p} \),

\[ (PS.2) \text{the graph} \]

\[ G_0 + \sum_{i=1}^{k} \tilde{G}_i \]

\[ \text{is a } (1 + \hat{\epsilon})^{1+\log_{12/11} \text{Vol}(V)} \text{ approximation of } G, \text{ and} \]

\[ (PS.3) \text{the total number of edges in } \tilde{G}_1, \ldots, \tilde{G}_k \text{ is at most } c_3(\log^{26}(n/p)/\hat{\epsilon}^2)|V|, \text{ for some constant } c_3. \]

Proof. We first observe that whenever the algorithm calls itself recursively, the volume of the graph in the recursive call is at most \( 11/12 \) of the volume of the original graph. So, the recursion depth of the algorithm is at most \( \log_{12/11} \text{Vol}(V) \). 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 \texttt{Sample} in line 2, \( \tilde{G}_1 \) is a \((1 + \hat{\epsilon})\) approximation of \( G \) and the number of edges in \( \tilde{G}_1 \) satisfies \((S.2)\),

2. for every call to \texttt{Sample} in line 3a, \( \tilde{G}_1 + G(D) + \partial(D, V - D) \) is a \((1 + \hat{\epsilon})\) approximation of \( G \) and the number of edges in \( \tilde{G}_1 \) satisfies \((S.2)\), and

3. For every call to \texttt{ApproxCut} in line 1 for which the set \( D \) returned satisfies \( \text{Vol}(D) \leq (1/12)\text{Vol}(V) \), there exists a set \( W \) containing \( V - D \) for which \( \Phi_W^G \geq f_2(\phi) \), where \( f_2 \) was defined in \((11)\).

First observe that at most \( n \) calls are made to \texttt{Sample} and \texttt{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, then Theorem 4.1 tells us that the smallest non-zero normalized Laplacian eigenvalue of \( G(W) \) is at least \( \lambda \), where \( \lambda \) is set in line 0. So, the assumptions of Theorem 6.1 will hold each time \texttt{Sample} is called, and so assumptions 1 and 2 fail with probability at most \( n\hat{p} \) each. Thus, all three assumptions hold with probability at least \( 1 - 3n\hat{p} \).

Property \((PS.3)\) 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 \( \tilde{G}_1, \ldots, \tilde{G}_k \) be output of PartitionAndSample on \( G \), and let \( V_1, \ldots, V_k \) be the vertex sets of \( \tilde{G}_1, \ldots, \tilde{G}_k \), respectively. We will prove by induction on \( d \) that

\[ \sum_{i=1}^{k} \tilde{G}_i + \partial(V_1, \ldots, V_k) \text{ is a } (1 + \hat{\epsilon})^{d+1}-\text{approximation of } G. \]
We base our induction on the case in which the algorithm does not call itself, in which case it either returns the original graph in step 2, or it returns the output of Sample in line 3. In the first case, the assertion is trivial; and in the later case it 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 $Vol(D) \leq (1/12)Vol(V)$. In this case, let $H = G(D)$, let $\tilde{H}_1, \ldots, \tilde{H}_k$ be the graphs returned by the recursive call to PartitionAndSample on $H$, and let $W_1, \ldots, W_k$ be the vertex sets of $\tilde{H}_1, \ldots, \tilde{H}_k$. 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

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

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

$$G = G(V-D) + H + \partial(V-D, D)$$

$$\preceq (1 + \hat{\epsilon}) \left( \tilde{G}_1 + H + \partial(V-D, D) \right)$$

by assumption 2,

$$\preceq (1 + \hat{\epsilon}) \left( \tilde{G}_1 + (1 + \hat{\epsilon})^d \left( \sum_{i=1}^k \tilde{H}_i + H_0 \right) + \partial(V-D, D) \right)$$

by induction,

$$\preceq (1 + \hat{\epsilon})^{d+1} \left( \tilde{G}_1 + \sum_{i=1}^k \tilde{H}_i + H_0 + \partial(V-D, D) \right)$$

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

One may similarly prove

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

establishing (18) for $G$.

We now consider the case in which $Vol(D) > (1/12)Vol(V)$. In this case, let $H = G(D)$ and $I = G(V-D)$. Let $W_1, \ldots, W_k$ be the vertex sets of $\tilde{H}_1, \ldots, \tilde{H}_k$ and let $U_1, \ldots, U_j$ be the vertex sets of $\tilde{I}_1, \ldots, \tilde{I}_j$. By our inductive hypothesis, we may assume that $\partial(W_1, \ldots, W_j) + \sum_{i=1}^k \tilde{H}_i$ is a $(1 + \hat{\epsilon})^d$-approximation of $H$ and that $\partial(U_1, \ldots, U_j) + \sum_{i=1}^j \tilde{I}_i$ is a $(1 + \hat{\epsilon})^d$-approximation of $I$. These two assumptions immediately imply that

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

is a $(1 + \hat{\epsilon})^{d+1}$-approximation of $G$, establishing (18) in the second case.

As the recursion depth of this algorithm is bounded by $\log_{12/11} Vol(V)$, we have established property (PS.2). \qed
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 \texttt{UnwtedSparsify}(\( G, \epsilon, p \)). Then,

\begin{enumerate}[(U.1)]
    \item The edges of \( \widetilde{G} \) are a subset of the edges of \( G \),
    \item with probability at least \( 1 - p \), \( \widetilde{G} \) is a \((1 + \epsilon)\)-approximation of \( G \), and
    \item \( \widetilde{G} \) has at most \( c_4 n \log^{27} (n/p) / \epsilon^2 \) edges, for some constant \( c_4 \).
\end{enumerate}

Moreover, the expected running time of \texttt{UnwtedSparsify} is at most \( O \left( m \log(1/p) \log^{11} n \right) \).

Proof. From (PS.1), we know that the depth of the recursion of \texttt{UnwtedSparsify} on \( G \) is at most \( \log_2 \text{Vol}(V) \leq 2 \log n \). 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 \texttt{PartitionAndSample} every time it is called by \texttt{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 \texttt{UnwtedSparsify}. We prove claim (U.2) by induction on the recursion depth. In particular, we prove that if \texttt{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 \texttt{UnwtedSparsify} makes no recursive calls to itself, in which case it returns at line 1 with a 1-approximation.

For \( d > 0 \), we assume 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 an \( (1 + \epsilon)^{(1 + \log_{12/11} n^2)} \leq (1 + \epsilon \ln 2 / (2 \log n)) \)

approximation of \( G \), as \( \epsilon \ln 2 / (2 \log n) \leq 1 \). By following the arithmetic in the proof of Lemma 9.1, we may prove that \( G_0 + \sum_{i=1}^{k} \widetilde{G}_i \) is a \((1 + \epsilon / (2 \log n + 1))^d\) approximation of \( G \).

To finish, we observe that

\[ (1 + \epsilon \ln 2 / (2 \log n))^{2 \log n} \leq 1 + \epsilon, \]

for \( \epsilon < 1 \).

Claim (U.1) follows from the observation that the set of edges of the graph output by \texttt{Sample} is a subset of the set of edges of its input.

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

Another multiplicative factor of \( O(\log n) \) comes from the logarithmic number of times that \texttt{UnwtedSparsify} can call itself during the recursion. \( \square \)
10 Sparsifying Weighted Graphs

In this section, we show how to sparsify graphs with edges of 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.

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.

\[ \tilde{G} = \text{BoundedSparsify}(G, \epsilon, p), \quad G = (V, E, w) \text{ has integral weights between 1 and } 2^u. \]

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

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

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

with probability at least \( 1 - p \),

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

(B.3) \( \tilde{G} \) has at most \( c_4 n \log U \log^{27}(n/p)/\epsilon^2 \) edges.

Moreover, the expected running time of BoundedSparsify is at most \( O \left( m u \log(1/p) \log^{11} n \right) \).

Proof. Immediate from Lemma 9.2.

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 \( U = n \). We remark
that a similar idea was used by Benczur and Karger [BK96] to build 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 \( \pi(a) = i \) if \( a \in V_i \). We define the contraction of \( G \) under \( \pi \) to be the weighted graph \( H = (\{1, \ldots, k\}, F, z) \), where \( F \) consists of edges of the form \((\pi(a), \pi(b))\) for \((a, b) \in E\), 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 \((a, b) \in E\) for which \( \pi(a) = \pi(b) \) do not appear in the contraction.

Given a weighted graph \( \tilde{H} = (\{1, \ldots, k\}, \tilde{F}, \tilde{z}) \), we say that \( \tilde{G} = (V, \tilde{E}, \tilde{w}) \) is a pullback of \( \tilde{H} \) under \( \pi \) if
1. \( \tilde{H} \) is a contraction of \( \tilde{E} \) under \( \pi \), and
2. for every edge \((i, j) \in \tilde{F}, \tilde{E}\) contains exactly one edge \((a, b)\) for which \( \pi(a) = i \) and \( \pi(b) = j \).

**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 \( \pi \) be the map of the partition. Set \( E_0 = \emptyset \), \( G_0 = (V, E_0, w) \), \( E_1 = E - E_0 \), and \( G_1 = (V, E_1, w) \). Assume that for some \( \epsilon < 1/2 \) and \( c \geq 3 \),
1. each set of vertices \( V_i \) is connected by edges in \( E_1 \),
2. every edge in \( E_1 \) has weight at least \( c^2 n^3 \),
3. every edge in \( E_0 \) has weight 1,
4. \( \tilde{G}_0 \) is a pullback under \( \pi \) of a \((1 + \epsilon)\)-approximation of the contraction of \( G_0 \) under \( \pi \).

Then, \( \tilde{G}_0 + G_1 \) is an \( \alpha \)-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 [ST06].

**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,
\[
(u, v) \preceq (1/w_1 + \cdots + 1/w_k) F.
\]
Proof of Lemma 10.2. Let $H$ be the contraction of $(V, E_0, w)$ under $\pi$, and let $\tilde{H}$ be the $(1 + \epsilon)$-approximation of $H$ for which $\tilde{G}_0$ 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 $\tilde{F}$ be the analogous graph for $\tilde{H}$. Our analysis will go through an examination of the graphs

$I \overset{\text{def}}{=} F + G_1$ and $\tilde{I} \overset{\text{def}}{=} \tilde{F} + G_1$.

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

(c) $\tilde{I}$ is a $(1 + 1/c)$-approximation of $\tilde{G} + G_0$.

To prove claim (a), consider any edge $(a, b) \in E_0$. If $\pi(a) = \pi(b)$, then the graph $1/\text{cn}^2 G_1$ contains a path from $a$ to $b$ of length at most $n$ on which every edge has weight at least $cn$. So, by Lemma 10.3

\[ (a, b) \preceq (n/cn) \frac{1}{\text{cn}^2} G_1 = (1/c) \frac{1}{\text{cn}^2} G_1. \quad (19) \]

If $\pi(a) \neq \pi(b)$, then the graph $1/\text{cn}^2 G_1$ 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

\[ (a, b) \preceq (1 + 1/c) \left( f + \frac{1}{\text{cn}^2} G_1 \right). \quad (20) \]

and

\[ f \preceq (1 + 1/c) \left( (a, b) + \frac{1}{\text{cn}^2} G_1 \right). \quad (21) \]

As there are fewer than $n^2/2$ edges in $E_0$, we may sum (19) and (20) over all of them (as appropriate) to establish

\[ G_0 \preceq (1 + 1/c) \left[ F + \frac{1}{2c} G_1 \right], \quad \text{so} \]

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

\[ \preceq (1 + 1/c) \left[ F + G_1 \right], \]

as $c \geq 1$. The inequality

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

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

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

Part (c) is very similar to part (a). We first note that the sum of the weights of edges in $\tilde{F}$ 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 \(\tilde{G}_0\) of weight \(w\), there is a corresponding edge \((v_{\pi(a)}, v_{\pi(b)})\) of weight \(w\) in \(\tilde{F}\). 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 part (a), we have
\[
e \preceq (1 + 1/c) \left( f + \frac{w}{cn^2} \right),
\]
and
\[
f \preceq (1 + 1/c) \left( e + \frac{w}{cn^2} \right).
\]
Summing these inequalities over all edges in \(\tilde{E}_0\), adding \(G_1\) to each side, and recalling \(\epsilon \leq 1/2\) and \(c \geq 3\), we establish part (c).

\[\tilde{G} = \text{Sparsify}(G, \epsilon, p), \text{ where } G = (V, E, w) \text{ has all edge-weights at most } 1.\]

0. Set \(Q = [6/\epsilon], \ b = 6/\epsilon, \ c = 6/\epsilon, \ \hat{\epsilon} = \epsilon/6, \) and \(l = [\log_2 2bc^2n^3].\)

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 \(\tilde{G} = (V, E, z)\), and express
\[
\tilde{G} = \sum_{i \geq 0} 2^{-i} G^i,
\]
where in each graph \(G^i\) all edges have weight 1, and each edge appears in at most \([\log_2 2Q]\) of these graphs.

3. Let \(E^i\) be the edge set of \(G^i\). Let \(E^{\leq i} = \bigcup_{j \leq i} E^j\). For each \(i, \) let \(D^{\leq i}_1, \ldots, D^{\leq i}_{n_i}\) be the connected components of \(V\) under \(E^{\leq i}\).

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 \(C^i_1, \ldots, C^i_{k_i}\) be the sets of form \(D^{\leq i-1}_j \cap V^i\) 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 \(E^{\leq i-1}\)). Let \(W^i = \bigcup_j C^i_j\).
   c. Let \(\pi\) be the map of partition \(C^i_1, \ldots, C^i_{k_i}\), and let \(H^i\) be the contraction of \((W^i, E^i)\) under \(\pi\).
   d. \(\tilde{H}^i = \text{BoundedSparsify}(H^i, \hat{\epsilon}, p/\ln)\).
   e. Let \(\tilde{G}^i\) be a pullback of \(\tilde{H}^i\) under \(\pi\) whose edges are a subset of \(E^i\).

5. Return \(\tilde{G} = \sum_i 2^{-i} \tilde{G}^i.\)
Lemma 10.4. Let \( k_i \) denote the number of clusters described by \text{Sparsify} at step 4b. Then,
\[
\sum_i k_i \leq 2\ln.
\]

**Proof.** Let \( \eta_i \) 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, so
\[
\eta_i \leq \eta_{i-1} - k_i/2.
\]
As the number of clusters never goes negative and is initially at most \( n \), we may conclude
\[
\sum_i k_i \leq 2\ln.
\]

\( \square \)

**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, let \( \tilde{G} \) be the output of \text{Sparsify}(G, \epsilon, p).

(X.1) The edges of \( \tilde{G} \) are a subset of the edges of \( G \), and

with probability at least \( 1 - p \),

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

(X.3) \( \tilde{G} \) has at most \( c_5 n \log^{28} (n/p) \log(1/\epsilon)/\epsilon^2 \) edges, for some constant \( c_5 \).

Moreover, the expected running time of \text{Sparsify} is at most \( O(\mu \log(1/p) \log^{13} n) \).

**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 \( \tilde{H}^i \) can be pulled back to edges of \( E^i \). But, this follows from (B.1) and the fact that \( H^i \) is a contraction of \( E^i \).

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

From Lemma 10.4, we know that there are at most \( ln \) values of \( i \) for which \( k_i \geq 2 \), and so \text{BoundedSparsify} is called at most \( ln \) times. Thus, with probability at least \( 1 - p \), the output returned by every call to \text{BoundedSparsify} satisfies properties (B.2 – 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 \( \tilde{H}_i \), and therefore in \( \tilde{G}_i \), is at most
\[
c_4 k_i \log n^2 \log^{27} (k_i/(p/2ln))/\epsilon^2 \leq c_4 k_i \log^{28} (2ln^2/p)/\epsilon^2.
\]
Applying Lemma 10.4, we may prove that the number of edges in \( \tilde{G} \) is at most
\[
\sum_i c_4 k_i \log^{28} (2ln^2/p)/\epsilon^2 \leq c_4 2ln \log^{28} (2ln^2/p)/\epsilon^2 \leq c_5 n \log^{28} (n/p) \log(1/\epsilon)/\epsilon^2,
\]
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

\[
\sum_{i \geq 0} 2^{-i} F^i = 2\tilde{G}.
\]

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

\[
\tilde{G}^i + c^2 n^3 F^{i-l}
\]

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

\[
\sum_{i \geq 0} 2^{-i} \left( \tilde{G}^i + c^2 n^3 F^{i-l} \right) = \tilde{G} + c^2 n^3 \sum_i 2^{-i} F^{i-l} = \tilde{G} + 2c^2 n^3 2^{-l} \tilde{G}
\]

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

\[
\sum_{i \geq 0} 2^{-i} \left( G^i + c^2 n^3 F^{i-l} \right) = \hat{G} + c^2 n^3 \sum_i 2^{-i} F^{i-l} = \hat{G} + 2c^2 n^3 2^{-l} \hat{G}.
\]

Setting

\[
\beta \overset{\text{def}}{=} 2c^2 n^3 2^{-l} \leq 1/b,
\]

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

\[(1 + \tilde{\epsilon})(1 + 1/c)^2(1 + \beta)\]

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

\[(1 + \tilde{\epsilon})(1 + 1/c)^2(1 + \beta)(1 + 1/Q) \leq (1 + \epsilon/6)^5 \leq (1 + \epsilon),\]

for \( \epsilon < 1/2 \).

To bound the expected running time of \textbf{Sparsify}, we observe that the time of the computation is dominated by the calls to \textbf{BoundedSparsify} and the time required to actually form the graphs \( H^i \). The sets \( D_j^{\leq i} \) may be maintained using union-find, 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 \( O(|E^i| \log n) \). So, the time to form the graphs \( H^i \) can be bounded by

\[
O \left( \sum_i |E^i| \log n \right) = O(m \lfloor \log 2Q \rfloor \log n) = O(m \log(1/\epsilon) \log n).
\]

This is dominated by our upper bound on the time required in the calls to \textbf{BoundedSparsify}, which is

\[
O \left( \sum_{|E^i|} \log n \log(1/p) \log^{11} n \right) = O \left( m \log(1/\epsilon) \log n \log(1/p) \log^{11} n \right) = O \left( m \log(1/p) \log^{13} n \right).
\]

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

Proof. We have

$$\tilde{G} + \beta \hat{G} \preceq (1 + \gamma)(1 + \beta)\hat{G},$$

which implies

$$\tilde{G} \preceq (1 + \gamma)(1 + \beta)\hat{G}.$$

On the other hand,

$$(1 + \beta)\hat{G} \preceq (1 + \gamma)\left( \tilde{G} + \beta \hat{G} \right)$$

implies

$$(1 - \beta \gamma)\hat{G} \preceq (1 + \gamma)\tilde{G},$$

which implies

$$\hat{G} \preceq \frac{1 + \gamma}{1 - \beta \gamma} \tilde{G} \preceq (1 + \beta)(1 + \gamma)\tilde{G},$$

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

10.1 Bounding Blow-Up

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

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

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

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

The algorithm in [ST06] 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 $C_i^j$ with many edges of $E^j$ on their boundary. These clusters can become vertices of high degree in the contraction by $\pi$, and can 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 $C_i^j$ so all the vertices in each cluster have approximately the same degree, and so that the degree of every vertex in $H^j$ is at most four times the degree of the vertices that map to it. Then, we set $\tilde{G}_i$ to be a random pullback of $\tilde{H}_i$ whose edges are a subset of $E$. That is, for each edge $(c,d) \in \tilde{H}_i$ 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.
\( \tilde{G} = \text{Sparsify}^2(G, \epsilon, p) \), 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 \( \delta \), let \( \delta C^i_1, \ldots, \delta C^i_k \) be the sets of form \( D^{\leq i-l} \cap \delta V^i \) that are non-empty and have an edge of \( E^i \) on their boundary. Let \( W^i = \bigcup_j \delta C^i_j \). For each set \( \delta C^i_j \) that has more than \( 2^{\delta+2} \) edges of \( E^i \) on its boundary, sub-divide the set until each part has between \( 2^\delta \) and \( 2^{\delta+2} \) edges on its boundary. Let \( \delta C^i_1, \ldots, \delta C^i_k \) be the resulting collection of sets.

4c. Let \( \pi \) be the map of partition of \( W^i \) by the sets \( \{ \delta C^i_j \} \), and let \( H^i \) be the contraction of \( (W^i, E^i) \) under \( \pi \).

4e. Let \( \tilde{G}^i \) be a random pullback of \( \tilde{H}^i \) under \( \pi \) 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 \( \delta C^i_j \) has degree at most \( 2^\delta+1 \). So, if we greedily pull off vertices 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 my at most \( 2^\delta+1 \). Thus, at the point when the size of the boundary of the new set first exceeds \( 2^\delta \), the size of the boundary of the old set must be at least \( 2^\delta - 2^\delta - 2^\delta+1 \geq 2^\delta \). 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 \( \tilde{G} \) be the output of \( \text{Sparsify}^2(G, \epsilon, p) \). Then,

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

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

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

(Y.3) \( \tilde{G} \) has at most \( c_6 n \log^{29}(n/p) \log(1/\epsilon)/\epsilon^2 \) edges, for some constant \( c_5 \),

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

Moreover, the expected running time of \( \text{Sparsify}^2 \) is at most \( O \left( m \lg(1/p) \lg^{13} n \right) \).

**Proof.** To prove (Y.3), we must bound the number of clusters produced in the modified step 4b. From Lemma 10.4, we know that

\[
\sum_i k^\delta_i \leq 2\ln n. \quad (22)
\]

To bound \( \sum_i k^\delta_i \), let \( \partial_{E_i}(W) \) denote the set of edges in \( E_i \) leaving a set of vertices \( W \). Let \( S^\delta_i \) be the set of \( j \) for which \( \delta C^i_j \) was created by subdivision, and recall that for all \( j \in S^\delta_i \),

\[ \left| \partial_{E_i} \left( \delta C^i_j \right) \right| \geq 2^\delta. \]
So,
\[ \sum_{j \in S_i} |\partial E_i \left( \delta C^i_j \right)| \geq 2^\delta (k_i^\delta - k_i^\delta), \]
and
\[ \sum_{i,j \in S_i} |\partial E_i \left( \delta C^i_j \right)| \geq 2^\delta \sum_i (k_i^\delta - k_i^\delta). \]  
(23)

As vertices in \( \delta V \) have at most \( 2^{\delta+1} \) edges and each edge of \( \hat{G} \) only appears in at most \( \lceil \log 2Q \rceil \) sets \( E^i \),
\[ \sum_{i,j \in S_i} |\partial E^i \left( \delta C^i_j \right)| \leq \lceil \log 2Q \rceil 2^{\delta+1} |\delta V|. \]  
(24)

Combining (23) with (24) and (22), we get
\[ \sum_i \kappa_i^\delta \leq 2 \lceil \log 2Q \rceil |\delta V| + 2 \ln, \]
and so
\[ \sum_{\delta, i} \kappa_i^\delta \leq 2 \lceil \log 2Q \rceil n + 2 \ln \lceil \log 2n \rceil \leq c_8 n \log(n/\epsilon), \]
for some constant \( c_8 \). By now applying the analysis from 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 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 \texttt{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
\[ \Pr \left[ \text{blow-up}_{\hat{G}} (v) > 2 \right] \leq e^{-48 \log(3n/p)^2/3}. \]
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 \texttt{Sparsify2} is similar to the analysis of \texttt{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 \( \delta C^i_j \) 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

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

As

$$\partial_{E_i} \left( \delta^iC^i_j \right) \geq 2^\delta$$

whenever we subdivide $\delta^iC^i_j$, we have

$$\sum_{j \in S^\delta_i} \partial_{E_i} \left( \delta^iC^i_j \right) \geq 2^\delta |S^\delta_i|.$$

Now, by (24)

$$\sum_{i} 2^\delta |S^\delta_i| \leq [\log 2Q] 2^{\delta+1} |V| \leq 2[\log 2Q] \Vol \left( \delta V \right).$$

Thus,

$$\sum_{\delta,i} 2^{\delta+1} |S^\delta_i| \leq 4[\log 2Q] \Vol \left( \delta V \right) = O(m \log(1/\epsilon)).$$

The stated bound on the expected running time of $\text{Sparsify2}$ follows.

\[\square\]

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

$$\mathbb{E} \left[ \text{blow-up}_{\tilde{G}} (e) \right] \leq 1. \quad (25)$$

**Proof.** We first observe that

$$\mathbb{E} \left[ \text{blow-up}_{\tilde{G}} (e) \right] = 1. \quad (26)$$

holds for the graph $\tilde{G}$ output by $\text{Sample}$ 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 $\text{UnwtedSparsify}$ merely partitions its input into edge-disjoint subgraphs and then applies $\text{Sample}$ to some of them, (26) holds for the output of $\text{UnwtedSparsify}$ as well.

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

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

We have

$$w_e = \sum_{i} 2^i w^i_e.$$

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

$$\mathbb{E} \left[ \tilde{w}^i_e \right] = 1.$$
So,
\[
E[\text{blow-up}_{\tilde{G}}(e)] = E\left[\frac{\sum_i 2^i \tilde{w}_i^e}{w_e}\right] = \frac{\sum_i 2^i E[\tilde{w}_i^e]}{w_e} = \frac{\sum_i 2^i w_i^e}{w_e} = 1,
\]
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 \(G^i\) is at most 1. If \(e\) is not on the boundary of a set \(C_j^i\), then \(e\) will not appear in \(G^i\) 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 \(\tilde{H} = (Y, \tilde{F}, \tilde{y})\), then \(w_e = y(\pi(u), \pi(v))\).

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

\[\square\]

**Lemma 10.9.** Let \(\tilde{G} = (V, \tilde{E}, \tilde{w})\) be the graph output by Sparsify2 on input \(G = (V, E, w)\). Then, for every \((u, v) \in E\),
\[
\text{blow-up}_{\tilde{G}}(u, v) \leq \frac{\min(d_u, d_v)}{48 \log((3n/p)^2)}.
\]

**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
\[
\min(d_u, d_v)/(16 \log(3/p))^2.
\]

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 \(\pi\) satisfy
\[
d_v \geq 4d_a,
\]
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
\[
\text{blow-up}_{\tilde{G}_i^i}(u, v) \leq \frac{4 \min(d_u, d_v)}{(16 \log(3n/p))^2} = \frac{\min(d_u, d_v)}{48 \log((3n/p)^2)}.
\]

We now measure the blow-up of edges relative to \(\tilde{G}\) instead of \(G\), which can only over-estimate their blow-up. The lemma then follows from
\[
\text{blow-up}_{\tilde{G}}(u, v) = \sum_i 2^{-i} \text{blow-up}_{\tilde{G}_i}(u, v) \leq \frac{\min(d_u, d_v)}{48 \log((3n/p)^2)} \sum_i 2^{-i} \frac{z_{u,v}}{z_{u,v}} = \frac{\min(d_u, d_v)}{48 \log((3n/p)^2)}.
\]

\[\square\]
References

[ACL06] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. *focs*, 0:475–486, 2006.

[AM01] Dimitris Achlioptas and Frank McSherry. Fast computation of low rank matrix approximations. In ACM, editor, *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing: Hersonissos, Crete, Greece, July 6–8, 2001*, pages 611–618, New York, NY, USA, 2001. ACM Press.

[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, New York, USA, May 1996. ACM Press.

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

[BSS08] Joshua Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. Available at http://arxiv.org/abs/0808.0163, 2008.

[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.

[Che86] P Chew. There is a planar graph almost as good as the complete graph. In *SCG ’86: Proceedings of the second annual symposium on Computational geometry*, pages 169–177. ACM, 1986.

[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.
[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 *STOC ’08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 563–568, New York, NY, USA, 2008. ACM.

[ST03] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. available at http://arxiv.org/abs/cs.DS/0310051, 2003.

[ST06] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. Available at http://www.arxiv.org/abs/cs.NA/0607105, 2006.

[ST08] Daniel A. Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. In preparation, 2008.

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

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

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