SPECTRAL SPARSIFICATION FROM FIRST PRINCIPLES

by

Gifan Thadathil

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Honors Bachelor of Science in Computer Science with Distinction

Spring 2018

SPECTRAL SPARSIFICATION FROM FIRST PRINCIPLES

by

Gifan Thadathil

I certify that I have read this thesis and that in my opinion it meets the academic and professional standard required by the University as a thesis for the degree of Bachelor of Science.

Signed: _____

Sebastian Cioabă, Ph.D.
Professor in charge of thesis

Approved: _____

Felix Lazebnik, Ph.D.
Committee member from the Department of Mathematical Sciences

Approved: _____

Matthew DeCamp, Ph.D.
Committee member from the Board of Senior Thesis Readers

Approved: _____

Paul Laux, Ph.D.
Director, University Honors Program

ACKNOWLEDGMENTS

At first I would like to thank my thesis committee — Sebastian Cioabă, Felix Lazebnik, and Matthew DeCamp — for allowing me the opportunity to write this thesis. In particular, I want to acknowledge Dr. Cioabă for his mathematical insights and help over the whole process. Despite his insistence on "serious student" behavior, which I stubbornly refuse to exhibit, he makes an excellent advisor. At last, I would like to thank friends and family for their endless encouragement and support, namely in tolerating my disappearances.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Graphs are abstract mathematical objects used to model networks, and spectral graph theory is the subfield studying matrices, eigenvalues, and eigenvectors associated with graphs. We consider the spectral graph sparsification: the problem of constructing sparse approximations of dense graphs with respect to spectral characteristics. We provide an exposition of this problem from the ground up. We explore two motivations for this problem: as a means to contains data explosion and as a generalization of expander graphs. From here, we formalize the problem and provide three probabilistic algorithms to construct spectral sparsifiers. This is done with incremental improvement to demonstrate the intuitions and proof techniques underlying sparsification algorithms. The first algorithm does simple random sampling with fixed probability, and the other two algorithms make use of effective resistances, of which the latter employs a law of large numbers technique.

Chapter 1

A FIRST LOOK

1.1 Introduction

Graphs are combinatorial structures used to model networks, and more generally, relationships between objects. Graph theory is the subfield of mathematics studying these structures. In this thesis, we consider the problem of spectral graph sparsification in a principled manner. We shall make explicit, without loss of rigor, the motivations and intuitions behind this problem, the accompanying definitions, and some solutions. Along the way, we will comment on the proof techniques we employ.

Since graphs are abstract models, they are simple to define. Figure 1.1 gives the standard visual representation of a graph. The dots represent objects and the lines represent relationships between objects. A number of problems — scheduling

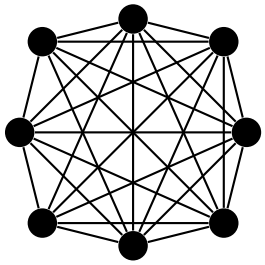
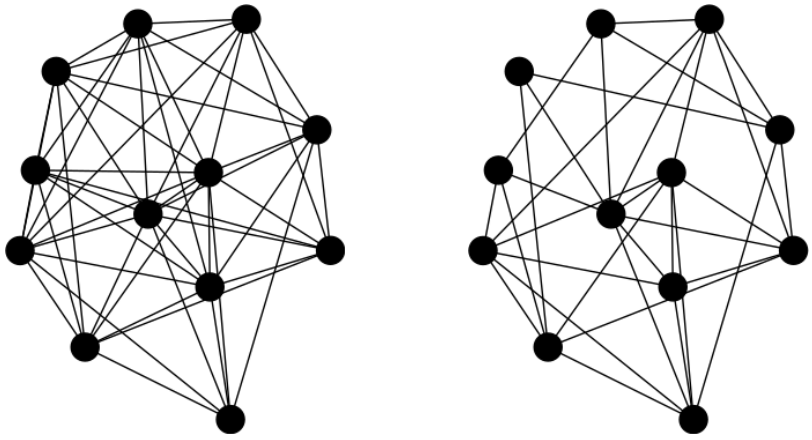


Figure 1.1: The **complete graph** (K_n) where $n = 8$. It is called complete since every possible edge is in the graph.

[10], route-finding [12], molecular dynamics [21], web search [20], and data clustering [27] to name a few — have graph-based solutions. As such, graphs are well studied objects in mathematics. Indeed, mathematicians have attacked graphs through non-graph theoretic means as well; algebraic, linear algebraic, probabilistic, and topological methods have all been used to tease out results.

The field of spectral graph theory studies graphs by looking at the eigenvalues and eigenvectors (generally referred to as the spectra) of their associated matrices. Surprisingly, this method of analysis can relate a graph's spectrum to various properties. If we take advantage of these facts, we arrive at solutions for some problems that commonly appear in computer science. We will focus on the problem of graph sparsification from the perspective of spectral graph theory.

Graph sparsification is the problem of approximating a dense graph with a sparse graph. For now, **sparse** loosely refers to having relatively few lines. Conversely, **dense** refers to having relatively many lines. Figure 1.2 visually demonstrates what we want our sparsification algorithm to do. When we want to sparsify graphs without changing its spectrum much, we call this problem **spectral graph sparsification**.



(a) The original graph. (b) Its sparse version.

Figure 1.2: An example sparsification of a dense graph by randomly sampling its edges with probabilities inversely proportional to the degrees of endpoints.

As for the remainder of this chapter, Section 1.2 will discuss a practical motivation for this problem. Section 1.3 then gives the technical background needed to understand the remainder of the thesis. Section 1.4 will define the problem more formally and define exactly what will constitute a solution. At the end of this chapter, Section 1.5 will provide a theoretical motivation for this problem. The second chapter will discuss algorithms to construct sparse graphs from the ground up, and the final

chapter will provide a summary of related papers for the interested reader.

1.2 A Means to Contain Data Explosion

There are two perspectives motivating spectral graph sparsification: an applied and theoretical perspective. To truly appreciate the theoretical perspective, we need some background, so we will defer that discussion to Section 1.5. The applied perspective, however, is in our grasp. From this perspective, a practitioner sparisfies graphs in order to mitigate the computational difficulties arising from data explosion.

Graphs are often treated as data structures in computer science, and so they are often explicitly or implicitly constructed during software development. In general, software will perform computations on these graphs. The runtime of these computations are necessarily dependent on the complexity of the graph. Here, "complexity" is a catch-all referring to the size of the vertex set, size of the edge set, the graph density, etc. Furthermore, the storage space for the graphs are inherently dependent on the graph's complexity as well. As the complexity of a graph increases, so does the runtime and storage space. If we are lucky, the increase is only linear or near-linear. If we are unlucky, the increases can be polynomial (with high degree) or even exponential. Of course, this does not matter if graphs are relatively simple.

Unfortunately this premise does not hold. Technological improvements have led to a so-called "Information Explosion." This has spurred the development of "Big Data," a loose term referring to giant, growing data sets too large for traditional techniques to handle, and the new techniques developed to address this deficiency [13]. It was estimated that in 2014, the world has the capacity to store 4.6 zettabytes [1] of (optimally compressed) information [15]. Since graphs are major data structures, they are only becoming more unmanageably complex, and computation speeds and storage space are becoming greater issues.

[1] This is roughly to 4284083843231 gigabytes!

These are remedied in part by increasing cloud computing/storage, GPU/parallel computing, etc. But that only addresses the problem at the hardware level. At the software level, the obvious way of addressing this problem is to develop better algorithms, perform optimizations, etc. Another option is to somehow reduce the complexity of graphs without too much loss in correctness. This is a sparsification problem. Indeed, for any algorithm operating on a graph, we can simply add sparsification as a preprocessing step as needed. In fact, spectral sparsification algorithms have been used to construct fast solvers for diagonally-dominant systems of linear equations [25]. Moreover, this was the initial motivation behind studying such algorithms.

One could argue that sparsification algorithms are unnecessary since many graphs appearing in the real-world are **small-world** graphs, which are very sparse to begin with [11]. However, we should be able to account for dense graphs on the off-chance we encounter one. After all, some algorithms involve the construction of dense graphs. For example, spectral clustering algorithms can construct the complete graph, which has maximum density. Therefore, we cannot get around needing to sparsify graphs. Indeed, spectral clustering performs well on spectral sparsified graphs [8]. We only need to sparsify fast enough to avoid substantial overhead.

1.3 Background

In this section, we give an overview of all technical background needed to understand the rest of the thesis. First, we will give some basic definitions about graphs. Then we will discuss three basic matrices we can associate with a graph: the degree, adjacency, and Laplacian matrices. Since they are all symmetric matrices, we will discuss some results about symmetric matrices next. Finally, we will discuss the Laplacian in greater detail.

1.3.1 Graphs

Formally, a **graph** is a pair $G = (V, E)$, where V is a set of vertices (dots) and E is a set of edges (lines). We will typically drop the "(V, E)" to simplify notation.

We will also typically refer to a vertex with a single letter, for example v, but vertices are indexed by natural numbers underneath. We will reserve the letter n to denote the size of the vertex set. An edge is a pair (u, v) connecting vertices u and v, which are called **endpoints** of the edge and **neighbors** of each other. When we do not need to refer to the endpoints, we will write an edge as e. We will reserve the letter m to denote the size of the edge set. The number of edges **incident** to, or connected to, a vertex is called the **degree** of that vertex. We will typically use d_v to refer to the degree of vertex v. If all vertices of a graph have degree d, then we call that graphs ***d*-regular**. We also say two vertices u, v are **adjacent** if they have an edge connecting them. In this case, we write it as $u \sim v$.

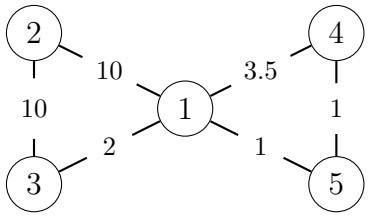


Figure 1.3: A weighted butterfly graph to exemplify the basic definitions for graphs. For example, 1 and 2 are adjacent vertices with an edge $(1, 2)$ with weight 10 connecting them. Furthemore, vertex 1 has degree 4.

For this thesis, we assume all graphs are simple. A **simple graph**, has no **loops** or **multiple edges**. A loop is an edge from a vertex to itself, and a graph has a multiple edge if it contains more than one edge between a pair of vertices. Although we restrict ourselves to simple graphs, we will consider **weighted graphs**. A weighted graph is a triple (V, E, w) where w is a function from E to $\mathbb{R}$, which associates a single real-valued, non-negative **weight** to each edge. We will use w_e or $w_{u,v}$ to denote the weight of an edge, and use $(u, v, w_{u,v})$ to refer to a weighted edge. An unweighted graph can be thought of as a weighted graph where all weights of edges are one and weights of non-edges are zero. If $G = (V, E)$ is a graph, and $G' = (V, E')$ is a graph where $E' \subseteq E$, then we say G' is a **spanning subgraph** of G. Finally, we say a graph G is **connected** if for every two vertices u, v, there exists a **path** from u to v through some sequence of edges existing in G.

1.3.2 Graphs and Matrices

There are three basic matrices that we will introduce first: the degree, adjacency, and Laplacian matrices. In the unweighted case, the degree matrix encodes the degrees of the vertices on diagonal elements, the adjacency matrix encodes when two vertices are adjacent, and the Laplacian combines both of these. In the weighted case, terms are simply multiplied by weights.

Note that we will use $M(i,j)$ to refer to the element of matrix M at row i and column j. To refer to a single row or column of M, we use $M(i,\cdot)$ and $M(\cdot,j)$ respectively. We give definitions for the more general weighted case since we switch between unweighted and weighted graphs. Figure 1.4 exemplifies these definitions with the unweighted butterfly graph. Finally, if we let G be a weighted graph with n vertices, we can define the matrices as follows.

Definition 1.1: *Degree Matrix*

The degree matrix of G is a $n \times n$ matrix D indexed by vertices where

$$D(i,j) = \begin{cases} \sum_{\ell:\ell\sim i} w_{i,\ell} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

where d_i is the degree of vertex i.

Definition 1.2: *Adjacency Matrix*

The adjacency matrix of G is a $n \times n$ matrix A indexed by vertices where

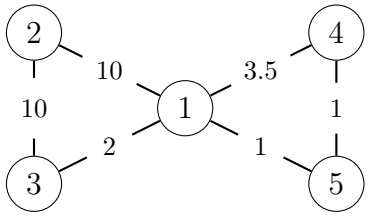
$$A(i,j) = \begin{cases} w_{i,j} & \text{if } i \sim j \\ 0 & \text{otherwise.} \end{cases}$$

Definition 1.3: *Laplacian Matrix*

The Laplacian matrix of G is a $n \times n$ matrix L indexed by vertices where

$$L(i,j) = \begin{cases} \sum_{\ell:\ell\sim i} w_{i,\ell} & \text{if } i = j \\ -w_{i,j} & \text{if } i \sim j \\ 0 & \text{otherwise.} \end{cases}$$

We can easily see that $L = D - A$.



$$D = \begin{bmatrix} 16.5 & 0 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 & 0 \\ 0 & 0 & 12 & 0 & 0 \\ 0 & 0 & 0 & 4.5 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} A = \begin{bmatrix} 0 & 10 & 2 & 3.5 & 1 \\ 10 & 0 & 10 & 0 & 0 \\ 2 & 10 & 0 & 0 & 0 \\ 3.5 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} L = \begin{bmatrix} 16.5 & -10 & -2 & -3.5 & -1 \\ -10 & 20 & -10 & 0 & 0 \\ -2 & -10 & 12 & 0 & 0 \\ -3.5 & 0 & 0 & 4.5 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix}$$

Figure 1.4: The butterfly graph with its degree, adjacency, and Laplacian matrices.

1.3.3 Symmetric Matrices

A **symmetric matrix** is a matrix M, for which $M = M^T$. The degree, adjacency, and Laplacian matrices are all symmetric. This is true for the degree matrix since it only has diagonal elements, and it is true for the other two since we consider undirected graphs (meaning (u, v) is an edge if and only if (v, u) is an edge). The fact that these matrices are symmetric gives us a whole body of useful results to take advantage of. One of the key facts about symmetric matrices we will use characterizes their eigenvalues and eigenvectors.

Theorem 1.1. [26] *If M is a real-valued $n \times n$ symmetric matrix, then it has n eigenvalues $\lambda_1, \cdots, \lambda_n$ with corresponding eigenvectors $u_1, \cdots, u_n$. Furthermore, $\lambda_1, \cdots, \lambda_n$ are all real numbers, and $u_1, \cdots, u_n$ form an orthonormal basis for $\mathbb{R}^n$.*

Theorem 1.1 is often given as a lemma to a final spectral theorem for symmetric matrices. We will not use it directly, but it is useful to know.

Theorem 1.2. [26] *If M is a real symmetric matrix, then M is **orthogonally diagonalizable**. That is, we can write $M = PDP^{-1}$ where D is a diagonal matrix, and P*

*is an **orthogonal matrix** (meaning $P^T = P^{-1}$). Furthermore, P is the matrix with eigenvectors of M making up the columns, and D has corresponding eigenvalues on the diagonal elements.*

The spectral theorem is useful because it allow us to derive the **eigendecomposition** of a symmetric matrix:

$$\begin{aligned} M &= PDP^{-1} \\ &= [\lambda_1 u_1, \cdots, \lambda_n u_n] P^{-1} \\ &\lambda_1 u_i u_1^T + \cdots + \lambda_n u_n u_n^T \\ &= \sum_{i=1}^{n} \lambda_i u_i u_i^T. \end{aligned} \tag{1.1}$$

This is a useful representation of symmetric matrices.

We can also associate a quadratic function to every $n \times n$ symmetric matrix M. This is called the **quadratic form** of M. It is a function q from $\mathbb{R}^n$ to $\mathbb{R}$ given by

$$q(x) = x^T M x = \sum_{i=1}^{n} \sum_{j=1}^{n} M(i,j) x_i x_j.$$

We can further categorize symmetric matrices by the values $x^T M x$ takes. We say that M is **positive semi-definite** if $x^T M x \geq 0$ for all x. Similarly, M is **negative semi-definite** if $x^T M x \leq 0$ for all x. If we enforce a strict $>$ or $<$, we say M is **positive definite** and **negative definite** respectively. In Proposition 1.7 we show the Laplacian is positive semi-definite, so we discuss such matrices in greater detail. In particular, we can show that positive semi-definite behavior transfers over to the eigenvalues as well.

Theorem 1.3. *If M is a positive semi-definite matrix, and $\lambda_1, \cdots, \lambda_n$ are its eigenvalues, then $\lambda_i \geq 0$ for all i.*

Proof. Let u_i be the eigenvector cooresponding to λ_i. Then

$$M u_i = \lambda_i u_i \implies \lambda_i = \frac{u_i^T M u_i}{u_i^T u_i}.$$

Since the numerator is non-negative by definitions, and the denominator is positive, it follows that $\lambda_i \geq 0$. ■

The quadratic form does more than just relate to the eigenvalues. In fact, we can write the smallest and largest eigenvalues as optimization problems on the **Rayleigh quotients** of M, which are given by $x^T Mx/x^T x$. Rayleigh quotients can be thought of as the quadratic form "normalized" by the length of x.

Theorem 1.4. *Let M be a $n \times n$ symmetric matrix with eigenvalues $\lambda_1 \leq \cdots \leq \lambda_n$. Then*

$$\lambda_1 = \min_{x \neq 0} \frac{x^T Mx}{x^T x} \quad \textit{and} \quad \lambda_n = \max_{x \neq 0} \frac{x^T Mx}{x^T x}.$$

Furthermore, the x obtaining the minimum and maximum are the eigenvectors of their respective eigenvalues.

Proof. We will show the proof for λ_1. The proof for λ_n follows from similar reasoning. We will prove this by showing $\lambda_1 \geq \min_{x \neq 0} x^T Mx/x^T x$ and $\lambda_1 \leq \min_{x \neq 0} x^T Mx/x^T x$. Let $u_1, \cdots, u_n$ be the eigenvectors for $\lambda_1, \cdots, \lambda_n$. If we plug in u_1 into the Rayleigh quotient and note $u_1 \neq 0$, we get

$$\frac{u_1^T Mu_1}{u_1^T u_1} = \frac{u_1^T \lambda_1 u_1}{u_1^T u_1} = \lambda_1.$$

It follows easily that

$$\lambda_1 \geq \min_{x \neq 0} \frac{x^T Mx}{x^T x}.$$

To show the other inequality, we note that since $u_1, \cdots, u_n$ form an orthonormal basis for $\mathbb{R}^n$, we can write

$$x = \sum_{i=1}^{n} c_i u_i$$

for all $x \in \mathbb{R}^n$ where $c_i \in \mathbb{R}$ or all i. Therefore,

$$\begin{aligned} \frac{x^T Mx}{x^T x} &= \frac{\left(\sum_{i=1}^n c_i u_i\right)^T M \left(\sum_{i=1}^n c_i u_i\right)}{\left(\sum_{i=1}^n c_i u_i\right)^T \left(\sum_{i=1}^n c_i u_i\right)} \\ &= \frac{\left(\sum_{i=1}^n c_i u_i\right)^T \left(\sum_{i=1}^n c_i M u_i\right)}{\sum_{i=1}^n \sum_{j=1}^n (c_i u_i)^T (c_j u_j)}. \end{aligned}$$

Since $u_1, \cdots, u_n$ are orthonormal, $u_i^T u_j = 0$ and $\|u_i\|^2 = 1$. Using this it follows

$$\begin{aligned}
\frac{x^T M x}{x^T x} &= \frac{\left(\sum_{i=1}^n c_i u_i\right)^T \left(\sum_{i=1}^n c_i \lambda_i u_i\right)}{\sum_{i=1}^n c_i^2} \\
&= \frac{\sum_{i=1}^n \sum_{j=1}^n (c_i u_i)^T (c_j \lambda_j u_j)}{\sum_{i=1}^n c_i^2} \\
&= \frac{\sum_{i=1}^n c_i^2 \lambda_i}{\sum_{i=1}^n c_i^2} \\
&\geq \frac{\lambda_1 \sum_{i=1}^n c_i^2}{\sum_{i=1}^n c_i^2} \\
&= \lambda_1.
\end{aligned}$$

Since the $x^T Mx/x^T x \geq \lambda_1$ for all x, we know

$$\min_{x \neq 0} \frac{x^T M x}{x^T x} \geq \lambda_1.$$

Thus we have $\lambda_1 = \min_{x \neq 0} x^T Mx/x^T x$ as desired. Equality for λ_n follows through similar reasoning. ■

Corollary 1.4.1. *For all $x \in \mathbb{R}^n$,*

$$\lambda_1 \leq \frac{x^T M x}{x^T x} \leq \lambda_n$$

with the bounds being met for $x = u_1$ and $x = u_n$ respectively.

We can extend these results further. Not only are the minimum and maximum eigenvalues given by optimizations on Rayleigh quotients; every eigenvalue is given by dual optimizations on Rayleigh quotients.

Theorem 1.5. *Courant-Fischer* [14]

Let M be a $n \times n$ symmetric matrix with eigenvalues $\lambda_1 \leq \cdots \leq \lambda_n$. If $S_i = \{ S \subseteq \mathbb{R}^n \mid dim(S) = i \}$, and $T_i = \{ T \subseteq \mathbb{R}^n \mid dim(T) = n - i + 1 \}$ are sets of i and $n - i + 1$ dimensional subspaces of $\mathbb{R}^n$ respectively, then

$$\lambda_i = \max_{S \in S_i} \min_{x \neq 0 \in S} \frac{x^T M x}{x^T x} = \min_{T \in T_i} \max_{x \neq 0 \in T} \frac{x^T M x}{x^T x}.$$

1.3.4 Properties of the Laplacian

The definition of the Laplacian given before is simple, but there is a more useful way to write the Laplacian.

Theorem 1.6. *The Laplacian matrix of G can be written as*

$$L = \sum_{(u,v)\in E} w_{u,v}(\chi_u - \chi_v)(\chi_u - \chi_v)^T$$

where χ_i is the ***characteristic vector*** *for vertex i containing 1 in the ith component and 0 everywhere else.*

Proof. We will only show one direction needed to prove equivalency to the original definition. We derive this definition from $L = D - A$. To do this, we will construct a matrix M and show $M = L$. Let M initially be the 0 matrix. Let us iterate over each edge (u, v) in G. We need to do two operations. First we set $M(u, v)$ and $M(v, u)$ to be $-w_{u,v}$. Then we add $w_{u,v}$ to $M(u, u)$ and $M(v, v)$. We do each operation for two elements in the matrix since an edge (u, v) is the same as (v, u). After doing this for every edge, we have $M = L$.

From this description, we know M must be the sum of matrices constructed by iterating over edges. That is,

$$M = \sum_{(u,v)\in E} M_{u,v}.$$

Each term M_{uv} in this sum will contain both operations for edge (u, v). We can perform both operations in one calculation with $(\chi_u - \chi_v)$. Note that $(\chi_u - \chi_v)$ will have 1 in coordinate u and -1 in coordinate v. If we let $M_{u,v}$ be the outer product shown in the theorem statement, we have

$$M_{u,v}(i,j) = [w_{i,j}(\chi_u - \chi_v)(\chi_u - \chi_v)^T](i,j) = \begin{cases} w_{u,v} & \text{if } i,j = u, \text{ or } i,j = v \\ -w_{u,v} & \text{if } i = u \text{ and } j = v \text{ or vice versa} \\ 0 & \text{otherwise.} \end{cases}$$

In (1.3.4), we set the diagonal elements $M_{u,v}(u,u)$ and $M_{u,v}(v,v)$ to be $w_{u,v}$, and the elements $M_{u,v}(u,v)$ and we set $M_{u,v}(v,u)$ to be $-w_{u,v}$ with 0 everywhere else. This is exactly what our two operations do. Thus we have

$$L = \sum_{(u,v)\in E} w_{u,v}(\chi_u - \chi_v)(\chi_u - \chi_v)^T.$$

■

We write the Laplacian in this manner to gain an insightful way of writing the quadratic form of the Laplacian:

$$\begin{aligned} x^T L x &= x^T \left[\sum_{(u,v)\in E} w_{u,v}(\chi_u - \chi_v)(\chi_u - \chi_v)^T \right] x \\ &= \sum_{(u,v)\in E} w_{u,v}(x_u - x_v)(x_u - x_v) \\ &= \sum_{(u,v)\in E} w_{u,v}(x_u - x_v)^2. \end{aligned}$$

Furthermore, since $x^T L x$ is a sum of squares, we know $x^T L x \geq 0$ for all x, so we know the following.

Proposition 1.7. *L is positive semi-definite.*

Still, we are not limited to the Laplacian representations given in Definition 1.3 and Theorem 1.6. We can also write the Laplacian in terms of the oriented incidence matrix.

Definition 1.4: *Oriented Incidence Matrix*
An oriented incidence matrix of a graph G is a $m \times m$ matrix M. The rows of M are indexed by edges and the columns are indexed by vertices. M is defined

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

where (i,j) is a directed edge and v is a vertex.

Theorem 1.8. *If L is the Laplacian matrix of a graph G, and M is the oriented incidence matrix of a graph, then $L = M^T W M$ where W is a $m \times m$ matrix with the edge weights on the diagonal elements.*

Proof. From the definition of matrix multiplication, we can write

$$[M^T W](i,j) = \sum_{\ell=1}^{m} M^T(i,\ell) W(\ell,j).$$

Using this we can write

$$\begin{aligned}[[M^T W]M](i,j) &= \sum_{k=1}^{n} [M^T W](i,k) M(k,j) \\ &= \sum_{k=1}^{n} \sum_{\ell=1}^{m} M^T(i,\ell) W(\ell,k) M(k,j).\end{aligned}$$

When $\ell \neq k$, we have $W(\ell,k) = 0$ since it is a diagonal matrix. Thus we only need to iterate over k:

$$[[M^T W]M](i,j) = \sum_{k=1}^{n} M^T(i,k) W(k,k) M(k,j).$$

Now we consider two cases: when $i = j$ and $i \neq j$. In the first case, we will get the diagonal elements of L, and in the second case, we will get the non-diagonal elements.

Suppose $i = j$, then $M^T(i,k) = M(k,j)$ by definition of transposition. Since $M(k,i)$ is 1 or -1 when vertex i is incident to edge k and 0 when it is not incident, we add 1 to the sum for each edge incident to i. Therefore $[[M^T W]M](i,j) = d_i$ when $i = j$.

Now suppose that $i \neq j$. If k is incident to only one of i or j, or if k is incident to neither, then $M^T(i,k)$ or $M(k,j)$ or both are 0, so we do not contribute to the sum. If k is incident to both i and j, then there exists an edge (i,j) in G. This means $M^T(i,k) = 1$ and $M(k,j) = -1$, or vice versa. In either case, we would add $-w_{i,j}$ to the sum. There are no multiple edges, so there can only be one value of k that produces the edge (i,j). Thus $[[M^T W]M](i,j) = -w_{i,j}$ when $i \sim j$. If no edge (i,j) exists, then $[[M^T W]M](i,j) = 0$. Putting both cases together, we notice that the entries of $M^T W M$ is identical to the entries of L. ■

We now present some basic results regarding the Laplacian's spectrum. First, we can relate the spectrum of the adjacency matrix to the spectrum of the Laplacian matrix in the regular case.

Theorem 1.9. *Let G be a d-regular graphs with adjacency matrix A and Laplacian matrix L. If $\lambda_1 \geq \cdots \geq \lambda_n$ are the eigenvalues of L with corresponding eigenvectors $u_1, \cdots, u_n$, then $d - \lambda_1 \geq \cdots \geq d - \lambda_n$ are the eigenvalues of A with the same eigenvectors.*

Proof. Let v_i be the eigenvector of ν_i. By definition, $Av_i = \nu_i v_i$. Recall the Laplacian can be defined as $L = D - A$. In the regular case, $D = dI$, so rearranging gives us $A = dI - L$. Applying this to the definition of an eigenvalue gives us

$$Av_i = \nu_i v_i \implies (dI - L)v_i = \nu_i v_i \implies dv_i - Lv_i = \nu_i v_i \implies Lv_i = (d - \nu_i)v_i.$$

This shows us that $d - \nu_i$ is an eigenvalue of L with eigenvector v_i. Because this holds for all i, the ith largest eigenvalue of A, which is ν_i will correspond to the ith smallest eigenvalue of L, which is λ_i. Thus $\lambda_i = d - \nu_i$ and $u_i = v_i$. ■

To get a better idea of how to calculate the Laplacian's eigenvalues, we will find them for the complete graph K_n.

Theorem 1.10. *Let K_n be the unweighted complete graph on n vertices, and let L be the Laplacian of K_n. If $\lambda_1, \cdots, \lambda_n$ are the eigenvalues of L, then $\lambda_1 = 0$ and $\lambda_2 = \cdots = \lambda_n = n$.*

Proof. First we will show that $\lambda_1 = 0$. Consider the all-ones vector $\mathbf{1} \in \mathbb{R}^n$ with 1 in every component. It follows that

$$L\mathbf{1} = \mathbf{0} = 0\mathbf{1}.$$

This holds because the ith component of $L\mathbf{1}$ has the sum of elements in row i of L. Row i sums to 0 since $L(i,i) = d_i$ while the d_i other elements in the row are -1.

Thus 0 is a eigenvalue of L with eigenvector $\mathbf{1}$. We know $\lambda_1 = 0$ because L is positive semi-definite and 0 is the smallest non-negative number.

Now let $\lambda \neq \lambda_1$ be one of the other eigenvalues, and let u be its eigenvector. Let $v = Lu$. Consider the ith component of v:

$$v_i = \sum_{j=1}^{n} L(i,j)u_j = L(i,i)u_i + \sum_{j \neq i} L(i,j)u_j.$$

Note that since we have K_n is $(n-1)$-regular, $L(i,i) = n-1$. Furthermore, each of $L(i,j) = -1$ since each edge exists in the graph. So now we have

$$\begin{aligned} v_i &= (n-1)u_i - \sum_{j \neq i} u_j \\ &= nu_i - u_i - \sum_{j \neq i} u_j \\ &= nu_i - \sum_{j=1}^{n} u_j. \end{aligned}$$

Since L is symmetric, it has an orthonormal basis, so $\mathbf{1}^T u = 0$, which means $\sum_{j=1}^{n} u_j = 0$. At last we have

$$v_i = nu_i - 0 = nu_i$$

which means $v = Lu = nu$. Therefore n is an eigenvalue of L, and because we let λ be any eigenvalue other than λ_1, without loss of generality it follows that $\lambda_2 = \cdots = \lambda_n = n$. ■

1.3.5 Laplacian Inverse

The section header is lie — the Laplacian does not have a proper inverse. This stems from the fact that 0 is one of its eigenvalues. But it is no issue. We will get around this by defining a pseudo-inverse, which can still be used to solve for x when we have $Lx = b$, albeit with some restrictions in place. Let L be a Laplacian matrix with eigenvalues $\lambda_1, \cdots, \lambda_n$ and respective eigenvectors $u_1, \cdots, u_n$. Recall (from equation 1.1) that the Laplacian exhibits the eigendecomposition. The decomposition for the

Laplacian is given below. Note that we ignore indexing from 1 since the first eigenvalue is 0

$$L = \sum_{i=2}^{n} \lambda_i u_i u_i^T$$

The definition of the Laplacian's pseudo-inverse [2] is a clever modification of the eigendecomposition. The intution behind it is that the eigenvalues of M^{-1} are the inverses of the eigenvalues of M whenever such an inverse exists.

Definition 1.5: *Laplacian Pseudo-inverse*

The (Moore-Penrose) pseudo-inverse of M is given by

$$L^+ = \sum_{i=2}^{n} \frac{1}{\lambda_i} u_i u_i^T.$$

Despite looking like an inverse, it is not readily apparent why L^+ should act like an inverse. This particular fact can be derived from discussion of kernel spaces, projection matrices, and orthogonal complements. The first fact we need to use is that L and L^+ have the same kernel spaces.

Theorem 1.11. *If L is the Laplacian matrix of a connected graph, and L^+ is its pseudo-inverse, then* $\ker(L) = \ker(L^+)$

Proof. Recall that the **kernel** and **image** of L are defined as follows:

$$\ker(L) = \{\, x \in \mathbb{R}^n \mid Lx = 0 \,\}$$
$$\text{im}(L) = \{\, Lx \mid x \in \mathbb{R}^n \,\}.$$

Let the eigenvalues of L be $\lambda_1 \leq \cdots \leq \lambda_n$ with corresponding eigenvectors $u_1, \cdots, u_n$. Consider its eigendecomposition:

$$L = \sum_{i=2}^{n} \lambda_i u_i u_i^T.$$

[2] The pseudo-inverse can actually be defined for any symmetric matrix since they all exhibit an eigendecomposition.

Recall that $\mathbf{1}$, the all-ones vector, is the eigenvector corresponding to λ_1, the only zero eigenvalue. This means that $\ker(L) = \text{span}(\mathbf{1})$. We want to show $\ker(L^+) = \text{span}(\mathbf{1})$ as well.

Let $x \in \text{span}(\mathbf{1})$, then we can write $x = c\mathbf{1}$ for some scalar c. It follows that

$$\begin{aligned} L^+x &= \sum_{i=2}^{n} \frac{1}{\lambda_i} u_i u_i^T x \\ &= \sum_{i=2}^{n} \frac{1}{\lambda_i} u_i u_i^T c\mathbf{1} \\ &= \sum_{i=2}^{n} \frac{c}{\lambda_i} u_i u_i^T \mathbf{1}. \end{aligned}$$

Due to the orthonormal nature of the eigenvectors L, we know $u_i^T\mathbf{1} = 0$, for all i. This means each term in the summation becomes zero, so

$$L^+x = 0.$$

This proves that $\text{span}(\mathbf{1}) \subseteq \ker(L^+)$. The argument for the other direction is the same, just ran in reverse. Take some non-zero x in the $ker(L^+)$, the summation terms must be 0. For this to be the case, it must be orthogonal to all the eigenvectors. Only something in $\text{span}(\mathbf{1})$ could satisfy that. ■

Next we need to show that the image and kernel of L are orthogonal complements.

Theorem 1.12. *If L is a Laplacian matrix, then* $\ker(L)$ *and* $\text{im}(L)$ *are orthogonal complements.*

Proof. Recall that we say two subsets $V, W \subseteq \mathbb{R}^n$ are **orthogonal complements** if, for all $v \in V$ and $w \in W$, it holds that $v^Tw = 0$. Let $x \in \ker(L)$, and let $x' \in \text{im}(L)$. By definition, $Lx = 0$ and $Ly = x'$ for some $y \in \mathbb{R}^n$. We want to show $x^Tx' = 0$. Since L is symmetric $L = L^T$, so

$$x^Tx' = x^TLy = (Ly)^Tx = y^TL^Tx = y^TLx = y^T0 = 0.$$

■

Now we can demonstrate the first inverse-like property of L^+. If M^{-1} is an inverse for matrix M, it means that $M^{-1}M = MM^{-1} = I$ by definition. We can show that L^+L and LL^+ exhibit something similar. If we try to calculate LL^+, we find that

$$\begin{aligned} LL^+ &= \left(\sum_{i=2}^{n} \lambda_i u_i u_i^T\right)\left(\sum_{i=2}^{n} \frac{1}{\lambda_i} u_i u_i^T\right) \\ &= \sum_{i=2}^{n}\sum_{j=2}^{n} \frac{\lambda_i}{\lambda_j} u_i u_i^T u_j u_j^T. \end{aligned}$$

Since the eigenvectors of L are orthogonal, when $i \neq j$, we have $u_i^T u_j = 0$, so we write

$$\begin{aligned} LL^+ &= \sum_{i=2}^{n}\sum_{j=2}^{n} \frac{\lambda_i}{\lambda_j} u_i u_i^T u_j u_j^T \\ &= \sum_{i=2}^{n} u_i \|u_i\|^2 u_i^T \\ &= \sum_{i=2}^{n} u_i u_i^T \end{aligned}$$

as the eigenvectors are all normal as well. Identical reasoning tells us that

$$L^+L = \sum_{i=2}^{n} u_i u_i^T$$

This demonstrates that we can cancel out the eigenvalue terms, but we are still left with the outer products of eigenvectors. This term looks nothing like I, but underneath it functions almost identically to the identity. This is why we call L^+ an inverse as well. The matrix L^+L acts like the identity because it is a projection matrix. We say a square $n \times n$ matrix M is a **projection matrix** if $M^2 = M$.

Theorem 1.13. *A projection matrix M is an identity for its image. That is, if $x \in \mathrm{im}(M)$, then $Mx = x$.*

Proof. Let $x \in \mathrm{im}(P)$, then $My = x$ for some $y \in \mathbb{R}^n$. Therefore

$$Mx = MMy = M^2 y = My = x.$$

■

We know L^+L is a projection matrix since

$$\begin{aligned}(L^+L)^2 &= \left(\sum_{i=2}^{n} u_i u_i^T\right)\left(\sum_{i=2}^{n} u_i u_i^T\right) \\ &= \sum_{i=2}^{n}\sum_{j=2}^{n} u_i u_i^T u_i u_i^T \\ &= \sum_{i=2}^{n} u_i \|u_i\| u_i^T \\ &= \sum_{i=2}^{n} u_i u_i^T\end{aligned}$$

just like before, so L^+L is the identity on $\text{im}(L^+L)$, but it is more as well. Through similar reasoning as given in Theorem 1.11, we can show that $\ker(L^+L) = \ker(L)$. Since the kernel and image are orthogonal complements and the matrices have the same dimensions, we have $\text{im}(L^+L) = \text{im}(L)$. Thus it follows that L^+L acts as the identity on $\text{im}(L)$ as well.

Finally, if we have $Lx = b$, then $b \in \text{im}(L)$. Since its in the image, b is orthogonal to $\mathbf{1} \in \ker(L)$. If $b \neq 0$, then x cannot be in $\ker(L)$. Because the union of the image and kernal becomes $\mathbb{R}^n$ and $x \in \mathbb{R}^n$, x must be in $\text{im}(L)$ as well. Since L^+L acts as the identity on $\text{im}(L)$, we know

$$Lx = b \implies L^+Lx = L^+b \implies x = L^+b.$$

Thus we can conclude that whenever $b \perp \mathbf{1}$, we can solve for x in $Lx = b$.

1.4 Formalizing the Problem

Formally, let G be a connected, unweighted, and simple graph. Also let $\widetilde{G}$ be a weighted **subgraph** of G. We will say that $\widetilde{G}$ is a **sparsifier** of G if $\widetilde{G}$ is sparse and approximate to G. [3] In this section, we will flesh out this definition by considering what it means to be "sparse" and "approximate."

So far, we have assumed that our sparsifier only has less edges than the original graph. It could very well be the case that a sparsifier contains fewer vertices as well.

[3] We will use the tilde notation to refer to the sparse versions of objects.

However, we will limit ourselves to former case for two reasons. In practice, it is often the case that we can say a relationship (edge) between two objects (vertices) is unimportant enough to disregard its existence, but rarely can we disregard an object's existence. Furthemore, there are few linear algebraic tools for comparing spectras of graphs with differing number of vertices. Indeed, if G and H have a different number of vertices, they will have a different number of eigenvalues and eigenvectors.

1.4.1 Defining "Sparse"

A connected sparse graph can be defined in various, similar ways. The sparsest connected graph is a **tree**, which has n vertices and $n-1$ edges. If we remove any edge from a tree, it becomes disconnected. Conversly, the least sparse graph is the complete graph on n vertices, which has an edge between every pair of vertices. This amounts to $\binom{n}{2}$ edges. So perhaps, we can define the sparsity of G by taking the ratio

$$\frac{m}{\binom{n}{2}} \text{ or } \frac{m}{n-1}.$$

Alternatively, we say G is sparse, if we can provide an upperbound on the number of edges it has. Perhaps, we can put an upperbound on the maximum or average degree. Another popular option is to show that the number of edges of G is $O(n)$ [4] . Any of these definitions work since they are all related, but we will settle with providing an upperbound on the number of edges since everything else depends on it.

Definition 1.6: *c-sparse*

We say a graph G is c-sparse if $m \leq c \leq \binom{n}{2}$. Note that c can be written as a function of the graph. For example we can say $c = kn$ for some integer k.

Historically, the goal is to produce sparse graphs where the number of edges is linear in the size of the vertices. That is, there should be $O(n)$ edges in the sparse

[4] A non-negative function $f(x)$ is $O(g(x))$ if there exists $x_0 \in \mathbb{R}$ and positive $k \in \mathbb{R}^n$ such that $f(x) \leq kg(x)$ for all $x \geq x_0$. This is a manner of characterizing asymptotic behavior of functions called Big-O notation.

graph. Once we have a concrete analysis of algorithms, we will abstract the complexity away and characterize the algorithms in terms of asymptotics.

1.4.2 Defining "Approximate"

This is a far more difficult question to answer. It has a variety of answers, and the correct one to choose — context dependent. This is due to the nature of graphs. For some mathematical objects, it is easy to define approximation. For example, one real number approximates another real number if they are close in value. As the complexity of the object increases, so do the number of ways we can define approximation. For example, if we wanted to approximate a vector x, we could say y approximates x if $\|y - x\|$ is small. Alternatively, we could say y approximates x if their Euclidean angle is small. Graphs have hidden complexity, and we can define approximation with respect to many properties. In any case, the notion of approximation we use should be appropriate for our use-case and capture something in our intuition.

Before, we discuss exactly what our property $P(G)$ is, we will formalize our notion of similarity. Because we want to compare $P(G)$ to $P(\widetilde{G})$, it will be useful to assume $P(G)$ and $P(\widetilde{G})$ are real numbers. This will allow us to use the usual comparison operations ($<$, $>$, etc). Furthermore, many graph properties are real numbers. As with integers, if $P(\widetilde{G})$ is closer in value to $P(G)$, then $\widetilde{G}$ is a better approximation of G. We can mathematically write this as

$$P(G) - a \leq P(\widetilde{G}) \leq P(G) + b$$

where a and b are positive and define bounds on how far away $P(\widetilde{G})$ can be from $P(G)$. This is often called **absolute error**. If we instead write

$$aP(G) \leq P(\widetilde{G}) \leq bP(G)$$

then we have **relative error**. The latter form of approximation is more prevalent, so we will use it as well. Indeed, the goal is to have $a = 1-\epsilon$ and $b = 1+\epsilon$ with $0 \leq \epsilon \leq 1$.

Historically, the first notion of approximation was explored by Chew [9]. He studied **graph spanners**, which approximate the shortest-path distances between any

two vertices in the original graph within a factor of two ($a = 1, b = 2$). Next, Benczúr and Karger studied **cut-sparsifiers**, which approximate the weight of cut-sets over all possible cuts [5]. The notion of sparsification we will consider, called **spectral sparsification**, are a generalization of cut-sparsifiers introduced by Spielman and Teng [24]. They do not separate the notions of approximation and sparse when discussing spectral sparsifiers as we have, so our definition of spectral approximation is a slight modification of the original definition.

Since we want to approximate the spectral characteristics of a graph, a good first choice for the approximation property is to approximate the eigenvalues of G. Perhaps, we can generalize, and try to approximate the entire image of the characteristic polynomial. We will do better, however, by approximating the image of the Laplacian quadartic form instead.

Definition 1.7:

Let G and H be graphs, and let L_G and L_H be their Laplacians respectively. We say that H is ***(a, b)-spectral approximation*** *to G if and only if for all $x \in \mathbb{R}^n$*

$$a(x^T L_G x) \leq x^T L_H x \leq b(x^T L_G x)$$

where a, b are constants with $b \geq a$. To simplify notation, this property is sometimes written as

$$aL_G \preceq L_H \preceq bL_G.$$

1.4.3 Spectral Approximation as Eigenvalue Approximation

At first, it is not obvious why we would consider the quadratic form to approximate spectral characteristics over directly approximating the eigenvalues. But if we recall Courant-Fischer (Theorem 1.5), it becomes rather intuitive.

Theorem 1.14. *Let G and H be graphs, and let L_G and L_H be their Laplacians respectively. If H is an (a, b)-spectral approximation of G, then*

$$a\lambda_i^G \leq \lambda_i^H \leq b\lambda_i^G$$

where λ_i^G and λ_i^H are the ith smallest eigenvalues of L_G and L_H respectively.

Proof. Since H is a (a, b)-spectral approximation of G

$$a(x^T L_G x) \leq x^T L_H x \leq b(x^T L_G x)$$

Dividing all the way through by $x^T x$ gives us

$$a\frac{x^T L_G x}{x^T x} \leq \frac{x^T L_H x}{x^T x} \leq b\frac{x^T L_G x}{x^T x} \tag{1.2}$$

for all $x \in \mathbb{R}^n - \{0\}$. Recall that Courant-Fischer (Theorem 1.5) tells us

$$\lambda_i^G = \max_{S \in S_i} \min_{x \in S - \{0\}} \frac{x^T L_G x}{x^T x}$$

where $S_i = \{\, S \subseteq \mathbb{R}^n \mid dim(S) = i \,\}$ is the set of i-dimensional subspaces of $\mathbb{R}^n$.

Consider some arbitrary $S \in S_i$. If x_S minimizes

$$\min_{x \in S - \{0\}} \frac{x^T L_G x}{x^T x}$$

then by (1.2), we have

$$\min_{x \in S - \{0\}} \frac{x^T L_H x}{x^T x} \leq \frac{x_S^T L_H x_S}{x_S^T x_S} \leq b\frac{x_S^T L_G x_S}{x_S^T x_S} = b \min_{x \in S - \{0\}} \frac{x^T L_G x}{x^T x}. \tag{1.3}$$

This tells us that the minimum Rayleigh quotient over S in L_H is less than or equal to b times the minimum Rayleigh quotient over S in L_G for all $S \in S_i$.

Now let u_i^H be the eigenvector corresponding to λ_i^H. By Courant-Fischer, we know there exists $S' \in S_i$ such that

$$\lambda_i^H = \frac{(u_i^H)^T L_H (u_i^H)}{(u_i^H)^T (u_i^H)} = \min_{x \neq \in S'} \frac{x^T L_H x}{x^T x}.$$

By (1.3) and Courant-Fischer, this means that

$$\lambda_i^H = \min_{x \in S'} \frac{x^T L_H x}{x^T x} \leq b \min_{x \in S - \{0\}'} \frac{x^T L_H x}{x^T x} \leq b \max_{S \in S_i} \min_{x \in S - \{0\}} \frac{x^T L_G x}{x^T x} = b\lambda_i^G.$$

This line of reasoning, which utilizes the max-min formulation of Courant-Fischer, gives us the right side of the inequality. Through similar reasoning with the min-max formulation of Courant-Fischer, we can get the left side of the inequality. ■

1.4.4 Spectral Approximation as a Generalization of Cut-Sparsifiers

We can also show that every spectral approximation of G is a cut-sparsifier of G as well. A **cut** of G is a partition of V into two sets S and $V - S$. Since we only need to choose S for the cut, we will typically refer to a cut by S. We have a **proper cut** if $0 < |S| < n$, meaning it is not the empty set. When iterating over all possible cuts, we will typically enforce $0 < |S| < n/2$, so we only consider proper cuts and avoid repeating partitions.

The **boundary** $E(S, V - S)$ is equal to the set of edges going between the two sets: $E(S, V - S) = \{ (u, v) \mid u \in S \text{ and } v \in V - S \}$. The **boundary size** of a cut is given by

$$\partial(S, V - S) = \sum_{(u,v) \in E(S, V-S)} w_{u,v}.$$

A cut-sparsifier approximates the boundary. That is, we say H is a (a, b)-cut-sparsifier of a graph G if for all cuts S

$$a\partial_G(S, V - S) \leq \partial_H(S, V - S) \leq b\partial_G(S, V - S).$$

We will show that we can get exactly this inequality when H is an (a, b)-spectral approximation of G.

Theorem 1.15. *If H is a (a, b)-spectral approximation of G, then H is a (a, b)-cut-sparsifier of G as well.*

Proof. Let S be a cut, and define the characteristic vector of S be a vector y where $y_i = 1$ if vertex $i \in S$ and $y_i = 0$ otherwise. Recall that

$$y^T L y = \sum_{(u,v \in E)} w_{u,v}(y_u - y_v)^2.$$

If $u, v \in S$ or $u, v \in V - S$, then $(y_u - y_v)^2$ is 0. If u and v are in different partitions, then $(y_u - y_v)^2$ is 1. But such an edge would be in the boundary of S. Therefore $y^T L y$ gives the boundary size of S.

Since H is a (a, b)-spectral approximation of G, it holds that

$$a(x^T L_G x) \leq x^T L_H x \leq b(x^T L_G x)$$

for all $x \in \mathbb{R}^n$. Since the characteristic vector y is in $\mathbb{R}^n$ it follows that

$$a\partial_G(S, V-S) \leq \partial_H(S, V-S) \leq b\partial_G(S, V-S).$$

■

This theorem demonstrates that spectral approximation is a stronger property [5] than that of cut-sparsifiers since we get a property that says something for every possible $x \in \mathbb{R}^n$, not just characteristic vectors.

Now that we have formalized notions of sparseness and approximation, we can define what it means to be a spectral sparsifier.

Definition 1.8: *Spectral Sparisfier*

We say that a graph $\widetilde{G}$ is a (a, b, c)-spectral sparsifier of a graph G if $\widetilde{E}$ is a (possibly reweighted) subset of E, $\widetilde{G}$ is an (a, b)-spectral approximation of G, and $\widetilde{G}$ is c-sparse.

1.5 A Generalization of Expander Graphs

In Section 1.2 we discussed why spectral sparsification is of practical interest. In this section, we will demonstrate that spectral sparsifiers generalize the notion of expander graphs, thereby being of theoretical interest as well.

Expanders are informally defined as highly connected, but sparse regular graphs [16]. They have heavy use in computer science applications. For example consider a simple network system between computers. We can model this network with a graph if we let computer systems be vertices and wires between them be edges. We want to send packets between systems quickly (highly connected), but we do not want to introduce direct lines between every two systems (sparse). An expander graph would give us a good layout for the wires in this case.

They can be formally defined in different ways depending on our choice of expansion parameter. The expansion parameter should be a graph property related to

[5] Interestingly, the theorem statement is worded to say that a spectral sparisifer is a special case of a cut sparsifier, but the approximation property is more general in the former than in the latter, so we are inclined to say the former generalizes the latter.

the connectivity of the graph. By giving k as a lower bound on the expansion parameter, we say that G is "highly connected." By forcing G to be d-regular, we can say that G is sparse. Of course, both of these statements only hold practically for appropriately high and low values of k and d respectively.

Definition 1.9:

*A d-regular graph G is an **expander graph** if it has expansion parameter at least c, for some choice of expansion parameter.*

To choose our expansion parameter, we first need an intuition of what it means for a graph to be highly connected. Let us consider the most connected of graphs: the complete graph K_n, which has edges between every two vertices. It would seem that the number of edges in the graph is a good expansion parameter. This is a naive choice, however, as most of the edges could be in only some part of the graph as in Figure 1.5. This graph has 10 vertices, but is 3-regular, so it is fairly sparse. But it does not appeal to our intuition about highly connected graphs. It is highly connected left and right of the center, but the left most vertices are not easily connected to the right most vertices. This is due to the single center edge which acts as a bottleneck. The graph would be better connected if we removed an edge from the left and right parts, and added an edges between the parts, so we have more than a single edge going between the parts.

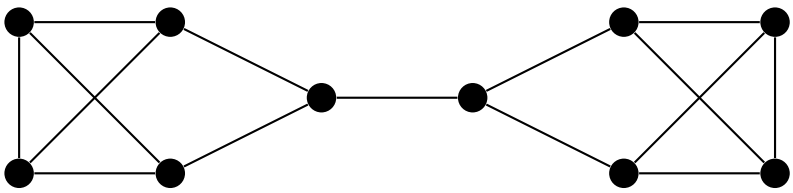


Figure 1.5: A graph with a bottleneck.

This notion of parts is the same as our notion of cuts discussed in Section 1.4.4. A definition for our expansion parameter would be to look at how much "bottleneck factor" appears in the graph. The original and popular definition for the expansion property is the isoperimetric number.

Definition 1.10:

*The **isoperimetric number** (also called the **Cheeger constant**) of a graph G is given by*

$$h = \min_{0<|S|\leq \frac{n}{2}} \frac{|E(S, V-S)|}{|S|}.$$

We can think of the isoperimetric number as the number of edges in the most restrictive bottleneck between two parts of G. The isoperimetric number of K_n is $n-1$, so G is more connected if h is closer to $n-1$. Despite its intutive definition, computational issues can arise when using it. The problem of computing the isoperimetric number is in CO-NP-Hard [16], which means it is NP-Hard to determine whether a number is not the isoperimetric number of a graph. That is good reason to believe its NP-Hard to compute the isoperimtric number. To get around this, we can define our expansion parameter to be the spectral gap.

Definition 1.11:

*Let G be a graph with adjacency matrix A, and let $\nu_1 \geq \nu_2 \geq \cdots \geq \nu_n$ be the eigenvalues of A. The **spectral gap** of G is defined as $\nu_1 - \nu_2$. If G is d-regular, then $\nu_1 = d$, so the spectral gap is $d - \nu_2$.*

The spectral gap is used to approximate the isoperimetric number due to the following theorem, which shows that the spectral gap can be used to provide lower and upper bounds to the isoperimetric number.

Theorem 1.16. *Alon-Milman [3] Let G be a graph with adjacency matrix A, and let $\nu_1 \geq \nu_2 \geq \cdots \geq \nu_n$ be the eigenvalues of A. If h is the isoperimetric number of G, then*

$$\frac{d-\nu_2}{2} \leq h \leq \sqrt{2d(d-\nu_2)}.$$

The advantage of the spectral gap is that it is written in terms of the eigenvalues, which are known to be computable in polynomial time by a number of algorithms. The problem of constructing expander graphs has been well-studied (see [16] for a survey on expanders), but it is still an open problem to construct the best expanders of arbitrary degree. These "best expanders" are called Ramanujan Graphs.

Definition 1.12:

A d-regular graph G with adjacency matrix eigenvalues $d = \nu_1 \geq \cdots \geq \nu_n \geq -d$ is a ***Ramanujan graph*** *if $\nu(G) \leq 2\sqrt{d-1}$ where $\nu(G) = \max_{1<i<n}|\nu_i|$.*

We will demonstrate how spectral sparsification is more general notion than expansion by showing that Ramanujan graphs are spectral sparsifiers for the complete graph. If we find algorithms to construct spectral sparsifiers, then we can sparsify arbitrary graphs.

Theorem 1.17. *If $\widetilde{G}$ is a Ramanujan graph with n vertices where each edge has weight n/d, then $\widetilde{G}$ is an (a, b, c)-spectral sparsifier for K_n where*

$$a = 1 - (2\sqrt{d-1}/d)$$
$$b = 1 + (2\sqrt{d-1}/d)$$
$$c = nd/2.$$

Proof. The Laplacian eigenvalues of K_n are 0 with multiplicity 1 and n with multiplicity $n-1$, so

$$0 \leq x^T L x \leq n$$

for all $x \in \mathbb{R}^n$. Let us momentarily assume, for the sake of argument, that $\widetilde{G}$ is unweighted. In this setting, we let $\nu_1 \geq \cdots \geq \nu_n$ be the eigenvalues of $\widetilde{A}$. Therefore $|\nu_i| \leq 2\sqrt{d-1}$ for all i as $\widetilde{G}$ is a d-regular Ramanujan graph. We can also write this as

$$-2\sqrt{d-1} \leq \nu_i \leq 2\sqrt{d-1}. \tag{1.4}$$

Therefore, if $\lambda_1 \leq \cdots \leq \lambda_n$ are the Laplacian eigenvalues of $\widetilde{G}$, then

$$d - 2\sqrt{d-1} \leq \lambda_i \leq d + 2\sqrt{d-1} \tag{1.5}$$

still assuming that $\widetilde{G}$ is unweighted.

Now we continue with the original assumption that $\widetilde{G}$ is weighted and all the weights are n/d. In this setting,

$$x^T \widetilde{L} x = \sum_{(u,v\in\widetilde{E})} w_{u,v}(x_u - x_v)^2 = \frac{n}{d} \sum_{(u,v\in\widetilde{E})} (x_u - x_v)^2 = \frac{n}{d}(x^T \widetilde{U} x)$$

where $\widetilde{U}$ is the Laplacian for $\widetilde{G}$ when it is unweighted. We already looked at the Laplacian eigenvalues in the unweighted setting; by (1.5), we know

$$d - 2\sqrt{d-1} \leq x^T \widetilde{U} x \leq d + 2\sqrt{d-1}$$

for all $x \in \mathbb{R}^n$. For the weighted case we only move the interval by the constant n/d, so we distribute $1/d$ to get

$$n\left(1 - \frac{2\sqrt{d-1}}{d}\right) \leq x^T \widetilde{L} x \leq n\left(1 + \frac{2\sqrt{d-1}}{d}\right)$$

for all $x \in \mathbb{R}^n$. We can rewrite this as

$$\left(1 - \frac{2\sqrt{d-1}}{d}\right) x^T L x \leq x^T \widetilde{L} x \leq \left(1 + \frac{2\sqrt{d-1}}{d}\right) x^T L x.$$

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

Since $\widetilde{G}$ is d-regular, it has $nd/2$ edges, so $\widetilde{G}$ is $(nd/2)$-sparse. It is also necessarily true that the edges of $\widetilde{G}$ are a subset of K_n since K_n has every possible edge. Putting it all together, we can conclude that $\widetilde{G}$ is a $(1 - (2\sqrt{d-1}/d), 1 + (2\sqrt{d-1}/d), nd/2)$-spectral sparsifier for K_n. ■

Chapter 2

ALGORITHM DESIGN

2.1 A First Algorithm

In the previous sction, we formed a well-defined notion of what it means for a graph to be a spectral sparsifier of another graph. Now we want to develop algorithms to construct spectral sparsifiers of arbitrary graphs. In this chapter, we start with a very simple algorithm for spectral sparsification, demonstrate some proof techniques, and analyze this first procedure to understand what considerations we should undertake to design a better algorithm.

Since sparsification amounts to choosing certain edges of a graph, the easiest way to sparsify a graph is to sample the edges with some probability. In our case, we will look at each edge of G and choose to include it in $\widetilde{G}$ with probability p. Intuitively, this should get us a sparse graph (for appropriately low values of p). However, this does not necessarily get us a good spectral sparsifier. Recall that the quadratic form of the Laplacian can be written as

$$x^T L x = \sum_{(u,v)\in E} w_{u,v}(x_u - x_v)^2.$$

If we only add some of the edges from G to $\widetilde{G}$, it is easy to see that $x^T \widetilde{L} x$ will have fewer terms in the sum and thus be smaller. But a spectral sparsifier approximates, not underestimates, $x^T L x$ with $x^T \widetilde{L} x$. To make up for this, we reweight the edges we do keep in $\widetilde{G}$. This bumps up the values contributed to the sum by the edges we do keep. We can derive exactly what choice of weight we should use from trying to satisfy certain properties.

In particular, we want to ensure that we have nice expectations in the quadratic form. That is, we want $\mathrm{E}\left[x^T \widetilde{L} x\right] = x^T L x$ for all x. If we treat the new weights $w_{u,v}$ as random variables, it follows from linearity of expectations that

$$\mathrm{E}\left[x^T \widetilde{L} x\right] = x^T L x \iff \sum_{(u,v)\in\widetilde{E}} \mathrm{E}\left[w_{u,v}\right](x_u - x_v)^2 = \sum_{(u,v)\in E} (x_u - x_v)^2$$

This means we need $\mathrm{E}\left[w_{u,v}\right] = 1$. This requirement is easily satisfied by reweighting the edges of G with the following scheme:

$$w_{u,v} = \begin{cases} \frac{1}{p} & \text{if we choose } (u,v) \text{ with probability } p \\ 0 & \text{otherwise} \end{cases}.$$

The algorithm in full detail is given in Algorithm 1. It iterates over each edge in G and chooses whether or not to include it in $\widetilde{G}$ with probability p. We will prove its correctness, but first we need to introduce Chernoff bounds, a proof technique essential to our analysis.

Algorithm 1 Basic Random Sampling

Require: $G = (V, E)$ is a connected, simple graph
Require: p is in $(0, 1)$
  function BASIC-RANDOM-SAMPLE(G, p)
    $\widetilde{E} \leftarrow \emptyset$
    $\widetilde{G} \leftarrow (V, \widetilde{E})$
    for all $(u, v) \in E$ **do**
      includeEdge? $\leftarrow$ *true* with probability p
      if *includeEdge*? **then**
        $\widetilde{G} \leftarrow (V, \widetilde{E} \cup (u, v, 1/p))$
      end if
    end for
    return $\widetilde{G}$
  end function

2.1.1 Chernoff Bounds

Chernoff bounds are an incredible tool of probability theory. They are used to bound the probability that a random variable given as the sum of other random

variables lies on the tail end of its distribution. Chernoff bounds can be derived by applying Markov's Inequality on the exponentiation of random variables.

Markov's inequality and Chernoff bounds are part of a family of theorems called "concentration inequalities" that allow one to bound the probability a random variables is far from some value. Proofs for the inequalities stated here and for many others can be found in [6]. We will introduce the basic bounds in detail since they are central to some of our arguments.

Theorem 2.1. *Markov's Inequality*

If X is a random variable taking non-negative values, and $a > 0$ is some constant, then

$$\Pr[X \geq a] \leq \frac{\mathrm{E}[X]}{a}.$$

Proof. For the sake of generality, we will assume X is continuous as well. In this case

$$\mathrm{E}[X] = \int_0^\infty x f(x) dx$$

where $f(x)$ is the probability density function for X. It follows that

$$\begin{aligned}
\mathrm{E}[X] &= \int_0^\infty x f(x) dx \\
&= \int_0^a x f(x) dx + \int_a^\infty x f(x) dx \\
&\geq \int_a^\infty x f(x) dx \\
&\geq \int_a^\infty a f(x) dx \\
&= a\Pr[X \geq a].
\end{aligned}$$

Thus we have

$$\Pr[X \geq a] \leq \frac{\mathrm{E}[X]}{a}.$$

■

Theorem 2.2. *Chernoff Bound on Bernoulli Variables*
Let $X_1, \cdots, X_n$ be independent random Bernoulli variables where $X_i = 1$ with probability p and $X_i = 0$ with probability $1 - p$. If $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathrm{E}[X] = np$, then for all $0 \leq \epsilon \leq 1$

$$\Pr[X \leq (1-\epsilon)\mu] \leq \exp\left(-\frac{\epsilon^2 \mu}{3}\right)$$

and

$$\Pr[X \geq (1+\epsilon)\mu] \leq \exp\left(-\frac{\epsilon^2 \mu}{3}\right).$$

Proof. We will only prove the part involving $(1+\epsilon)$. The other statement can be proved through a similar argument. To begin we observe that

$$\Pr[X \geq (1+\epsilon)\mu] = \Pr[\exp(tX) \geq \exp(t(1+\epsilon)\mu)]$$

for any $t \geq 0$. By Markov's inequality, we have

$$\Pr[X \geq (1+\epsilon)\mu] \leq \frac{\mathrm{E}[\exp(tX)]}{\exp(t(1+\epsilon)\mu)}. \tag{2.1}$$

Due to the definition of X and the independence of X_i, we can calculate the numerator with

$$\mathrm{E}[\exp(tX)] = \mathrm{E}[\exp(tX_1 + \cdots + tX_n)] = \prod_{i=1}^{n} \mathrm{E}[\exp(tX_i)].$$

Since X_i is a Bernoulli variable, it follows that

$$\mathrm{E}[\exp(tX_i)] = p(\exp(t)) + (1-p)\exp(0) = 1 + p(\exp(t) - 1).$$

It is a well known result that $1 + x \leq \exp(x)$ for any choice of x. Using this we can find that

$$\mathrm{E}[\exp(tX_i)] = 1 + p(\exp(t) - 1) \leq \exp(p(\exp(t) - 1)).$$

Combining the $\mathrm{E}\left[\exp(tX_i)\right]$, we can write

$$\begin{aligned}\mathrm{E}\left[\exp(tX)\right] &= \prod_{i=1}^{n} \mathrm{E}\left[\exp(tX_i)\right] \\ &\leq \prod_{i=1}^{n} \exp(p(\exp(t)-1)) \\ &= \exp(np(\exp(t)-1)) \\ &= \exp(\mu(\exp(t)-1))\end{aligned}$$

since $\mu = np$. Now that we have bounded $\mathrm{E}\left[\exp(tX)\right]$, we can further bound (2.1)

$$\Pr\left[X \geq (1+\epsilon)\mu\right] \leq \frac{\mathrm{E}\left[\exp(tX)\right]}{\exp(t(1+\epsilon)\mu)} \leq \frac{\exp(\mu(\exp(t)-1))}{\exp(t(1+\epsilon)\mu)}. \tag{2.2}$$

So far we have related the probability that X deviates from its expected value to some exponential. But we did this by introducing some variable t. In order to get the bound on the probability to be as tight as possible, we need to find t that minimizes the upper bound. The minimum is given by $t = ln(1+\epsilon)$. The proof of this particular fact is beyond our scope, but it is straightforward task in optimization. Substituting this value of t in (2.2) gives us

$$\begin{aligned}\Pr\left[X \geq (1+\epsilon)\mu\right] &\leq \frac{\exp(\mu(\exp(\ln(1+\epsilon))-1))}{\exp(\ln(1+\epsilon)(1+\epsilon)\mu)} \\ &= \frac{\exp(\mu\epsilon)}{(1+\epsilon)^{(1+\epsilon)\mu}} \\ &= \left(\frac{\exp(\epsilon)}{(1+\epsilon)^{(1+\epsilon)}}\right)^{\mu}.\end{aligned} \tag{2.3}$$

Often times Chernoff bound theorems are given with this sort of bound, but it is not particularly easy to use in practice. We will derive a looser, more convenient bound by finding a lower bound for $(1+\epsilon)\ln(1+\epsilon)$.

Consider the Taylor series expansion of $\ln(1+\epsilon)$:

$$\ln(1+\epsilon) = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{\epsilon^n}{i} \geq \epsilon - \frac{\epsilon^2}{2} + \frac{\epsilon^3}{3}.$$

We can substitute this into the original expression to get

$$\begin{aligned}(1+\epsilon)\ln(1+\epsilon) &\geq (1+\epsilon)(\epsilon - \frac{\epsilon^2}{2} + \frac{\epsilon^3}{3}) \\ &= \epsilon + (\epsilon^2 - \frac{\epsilon^2}{2}) + (\frac{\epsilon^3}{3} - \frac{\epsilon^3}{2}) + \frac{\epsilon^4}{3} \\ &\geq \epsilon + (\epsilon^2 - \frac{\epsilon^2}{2}) + (\frac{\epsilon^3}{3} - \frac{\epsilon^3}{2}) \\ &= \epsilon + \frac{\epsilon^2}{2} - \frac{\epsilon^3}{6}.\end{aligned}$$

If we also assume $0 \leq \epsilon \leq 1$, then $\epsilon^3 \leq \epsilon^2$, so

$$\begin{aligned}(1+\epsilon)\ln(1+\epsilon) &\geq \epsilon + \frac{\epsilon^2}{2} - \frac{\epsilon^3}{6} \\ &\geq \epsilon + \frac{\epsilon^2}{2} - \frac{\epsilon^2}{6} \\ &= \epsilon + \frac{\epsilon^2}{3}.\end{aligned}$$

Using this as a lower bound in (2.3), we finally have

$$\Pr\left[X \geq (1+\epsilon)\mu\right] \leq \left(\frac{\exp(\epsilon)}{\exp(\ln(1+\epsilon)(1+\epsilon))}\right)^{\mu} \leq \left(\frac{\exp(\epsilon)}{\exp(\epsilon + (\epsilon^2)/3)}\right)^{\mu} = \exp\left(-\frac{\epsilon^2\mu}{3}\right)$$

as desired. Through similar reasoning we can shown

$$\Pr\left[X \leq (1-\epsilon)\mu\right] \leq \exp\left(-\frac{\epsilon^2\mu}{3}\right)$$

■

We are not limited to the Chernoff bound given in Theorem 2.2. In reality, the term "Chernoff bounds" refer to a family of results all bounding the probabilities that random variables expressed as sums are on the tail ends of their distributions. The differences lie in the assumed properties of the random variables and the values ϵ can take. Theorem 2.2 expects that X_i is a Bernoulli variable. We can extend it to let X_i take any real value between 0 and 1 and non-negative ϵ.

Theorem 2.3. *Chernoff Bound on Small Variables* [6]

Let $X_1, \cdots, X_n$ *be independent random variables taking values in* $[0, 1]$. *If* $X = \sum_{i=1}^{n} X_i$ *and* $\mu = \mathrm{E}[X]$, *then for all* $\epsilon \geq 0$

$$\Pr[X \leq (1-\epsilon)\mu] \leq \exp\left(-\frac{\epsilon^2 \mu}{2+\epsilon}\right)$$

and

$$\Pr[X \geq (1+\epsilon)\mu] \leq \exp\left(-\frac{\epsilon^2 \mu}{2+\epsilon}\right).$$

We can apply a rescaling argument [1] to produce an even more general statement. Suppose that X_i took values in $[0, \beta]$. If we let $Y_i = X_i/\beta$ and have $Y = \sum_{i=1}^{n} Y_i$, then we can apply Theorem 2.3 on Y and substitute with X/β to get

$$\Pr\left[\frac{X}{\beta} \leq (1-\epsilon)\mu\right] \leq \exp\left(-\frac{\epsilon^2 \mu}{2+\epsilon}\right)$$
$$\Pr\left[\frac{X}{\beta} \geq (1+\epsilon)\mu\right] \leq \exp\left(-\frac{\epsilon^2 \mu}{2+\epsilon}\right)$$

where $\mu = \mathrm{E}[Y]$. If we notice that $\mathrm{E}[Y] = \mathrm{E}[X]/\beta$, then we easily get a rescaled Chernoff bound after some substitution and rearrangement.

Theorem 2.4. *Chernoff Bound on Non-negative Variables*

Let $X_1, \cdots, X_n$ *be independent random variables taking values in* $[0, \beta]$. *If* $X = \sum_{i=1}^{n} X_i$ *and* $\mu = \mathrm{E}[X]$, *then for all* $\epsilon \geq 0$

$$\Pr[X \leq (1-\epsilon)\mu] \leq \exp\left(-\frac{\epsilon^2 \mu}{(2+\epsilon)\beta}\right)$$

and

$$\Pr[X \geq (1+\epsilon)\mu] \leq \exp\left(-\frac{\epsilon^2 \mu}{(2+\epsilon)\beta}\right).$$

2.1.2 Analysis

To analyze BASIC-RANDOM-SAMPLE, we will make use of Chernoff bounds. We need to be careful about our analysis, however. There are few aspects of Chernoff

[1] Rescaling arguments can often extend results that put constraints on objects in the hypothesis to analogous results with looser constraints. Although, the object x in question should be representable in the form cx where c is some scalar.

bounds that make it difficult to use. To begin, the bounds it gives are awkward. It is more beneficial to write the bounds in terms of the input graph. To do this, we must choose X and find β in a way that simplifies the resulting bounds. We can also bound the bounds until we get something in appropriate terms. We have to be careful though since Chernoff bounds typically give loose bounds to begin with since they hold over arbitrary random variables. Despite their troublesome behavior, they are still a useful theoretical tool. To demonstrate the tricks of Chernoff bounds, we prove the following guarantee on our simple algorithm.

Theorem 2.5. *Let G be a connected and unweighted simple graph, $2/3 \leq p < 1$ be some probability, and let m be sufficiently large. If $\widetilde{G}$ is the output of* BASIC-RANDOM-SAMPLE(G, p)*, then $\widetilde{G}$ is a $(0, 5, mp + 1)$-spectral sparsifier of G with probability at least $1/2$.*

Proof. We know BASIC-RANDOM-SAMPLE terminates since the algorithm simply iterates over all edges in G, of which there is a finite number. We also know $\widetilde{E}$ is a subset of E since we are choosing edges from E.

To show that $\widetilde{G}$ is a spectral approximation of G, we will use Chernoff bounds to bound the probability of whether $\widetilde{G}$ is *not* a spectral approximation of G. This naturally gives us a bound on the probability that $\widetilde{G}$ is a spectral approximation of G.

In particular, we want to use Chernoff bounds to show

$$(1 - \epsilon)x^T Lx \leq x^T \widetilde{L}x \leq (1 + \epsilon)x^T Lx$$

for some value of ϵ. This may seem odd because in the end we need to show

$$0 \leq x^T \widetilde{L}x \leq 9x^T Lx$$

and $\epsilon = 8$ gives us a lower bound of -7 not 0. Since $\widetilde{L}$ is positive semi-defininite, any value of $\epsilon \geq 1$ is useless with regards to the lower bound since 0 will always be a lower bound. It turns out that we need $\epsilon \geq 1$ to get even an upper bound, so we only consider the upper bound part of Chernoff bounds.

Given the way Chernoff bounds are written, it seems natural to select $x^T\widetilde{L}x$ as our random variable. Let $X = x^T\widetilde{L}x$. Since we can write

$$x^T\widetilde{L}x = \sum_{(u,v)\in E} w_{u,v}(x_u - x_v)^2$$

we will let $X_{u,v} = w_{u,v}(x_u - x_v)^2$. But notice that our choice is currently inappropriate. We can make $x^T\widetilde{L}x$ arbitrarily high by simply letting one component of x be arbitrarily high and setting all others zero. Therefore we will not be able to find an appropriate β and will not be able to apply Chernoff bounds.

To resolve this, we restrict ourselves to $x \in \mathbb{R}^n$ such that $\|x\|^2 = 1$. If we show $\widetilde{G}$ is a spectral approximation of G in this restricted domain, then we can apply a rescaling argument to generalize to the entire domain. From here, there are two tasks we need to complete before applying Chernoff bounds: first, we must calculate $\mathrm{E}\left[x^T\widetilde{L}x\right]$, and then we need to find an upper bound β for the values of $w_{u,v}(x_u - x_v)^2$. We have done the former task already. By reweighting the graph with probability $1/p$, we forced $\mathrm{E}\left[x^T\widetilde{L}x\right] = x^TLx$. The second task requires some more thought, but amounts to finding exactly how high $w_{u,v}(x_u - x_v)^2$ can become given our restriction on x. The highest $w_{u,v}$ can become is $1/p$, which happens when edge (u, v) is included in $\widetilde{G}$. To derive the highest $(x_u - x_v)^2$ can become, we need to

$$\text{maximize } (x_u - x_v)^2 \text{ subject to } \sum_{i=1}^{n} x_i^2 = 1.$$

Suppose there exists some $x_i \neq 0$ where $i \neq u \neq v$. This choice of x will never obtain the maximum since we can always get a bigger value for $(x_u - x_v)^2$ by setting $x_i = 0$ and adjusting x_u to be larger in order to satisfy $\sum_{i=1}^{n} x_i^2 = 1$. Thus we know the vector x obtaining the maximum must have x_u and x_v as the only non-zero coordinates. This allows us to simplify the task; now we only need to

$$\text{maximize } (x_u - x_v)^2 \text{ subject to } x_u^2 + x_v^2 = 1$$

With this problem, finding the maximum amounts to locating the point on the unit circle that maximizes the squared difference of its coordinates. This is obtained for $x_u = 1/\sqrt{2}$ and $x_v = -1/\sqrt{2}$. Thus we know

$$w_{u,v}(x_u - x_v)^2 \leq \frac{1}{p}\left(\frac{2}{\sqrt{2}}\right)^2 = \frac{4}{p2} = \frac{2}{p} = \beta.$$

After applying the Chernoff bounds given in Theorem 2.4, we discover

$$\Pr\left[x^T\widetilde{L}x \geq (1+\epsilon)x^TLx\right] \leq \exp\left(-\frac{\epsilon^2 p(x^TLx)}{(2+\epsilon)2}\right).$$

for all $\epsilon \geq 0$. If we remove x^TLx from the numerator, we can get the bounds into a better form:

$$\Pr\left[x^T\widetilde{L}x \geq (1+\epsilon)x^TLx\right] \leq \exp\left(-\frac{\epsilon^2 p}{(2+\epsilon)2}\right).$$

If $x^T\widetilde{L}x \geq (1+\epsilon)x^TLx$, then $\widetilde{G}$ is not a spectral approximation of G. If we take opposite probabilities, then we find

$$\Pr\left[x^T\widetilde{L}x \leq (1+\epsilon)x^TLx\right] \geq 1 - \exp\left(-\frac{\epsilon^2 p}{(2+\epsilon)2}\right). \tag{2.4}$$

Remember that (2.4) only applies for unit vectors x. To generalize to $y \in \mathbb{R}^n$, we apply a rescaling argument. Note that $y = cx$ for some scalar c and unit vector x. We know

$$0 \leq x^T\widetilde{L}x \leq (1+\epsilon)x^TLx.$$

with at least some probability. If we multiply throughout by c^2, it follows that

$$0 \leq y^T\widetilde{L}y \leq (1+\epsilon)y^TLy$$

with at least the same probability. Now that we have shown that spectral approximation is possible, we need to simplify the probability bound.

For certain values of ϵ and p, the bound becomes useless. For example, with $\epsilon = p = .5$, the bound becomes around .02, which merely tells us that it is possible for $\widetilde{G}$ to be a spectral sparsifier. Therefore, we need to restrict ourselves to useful values of ϵ and p. Furthermore, it will be best to write ϵ in terms of p, so we have

only one parameter affecting the algorithmic guarantees. Since p is proportional to sparseness, we want to keep p low, and since ϵ is inversely proportional to the quality of the approximation, we want to keep ϵ low as well. However, making ϵ large is the easiest way to arrive at a relatively large bound. This is because ϵ^2 resides in the numerator of the exponent — we have a balancing act with keeping ϵ neither too low nor too high. Let us write $\epsilon = kp$ for some constant k. With this choice of ϵ, increasing p will give us larger ϵ, but to ensure that we do not have to increase p too much, we can use k to bump up ϵ. Therefore, we have reduced our problem to finding an appropriate value for k.

Suppose we want $\widetilde{G}$ to be a spectral sparsifier with "better than random" probability, which is $1/2$. With this constraint, let us try some values of p and k. If $p = 2/3$ and $k = 5$, then the bound in (2.4) is a little over $1/2$. In this case $\epsilon = kp = 10/3 \leq 4$, so we can conclude that, with probability at least $1/2$, $\widetilde{G}$ is a $(0, 5)$-spectral approximation of G when $p \geq 2/3$.

Now that we have shown $\widetilde{G}$ is a spectral sparsifier of G, we only need to put an upper bound on the number of edges in $\widetilde{G}$. We will do this using the Chernoff bound given in Theorem 2.2. Let $X_{u,v}$ now be a random Bernoulli variable indicating whether edge (u, v) was chosen to be in $\widetilde{G}$. That is, $X_{u,v} = 1$ with probability p and 0 with probability $1 - p$. If we let X be the number of edges in $\widetilde{G}$, it follows that

$$X = \widetilde{m} = \sum_{(u,v)\in E} X_{u,v}.$$

Taking expectations, we can easily see $\mathrm{E}[X] = mp$. By Theorem 2.2, it follows that

$$\Pr[\widetilde{m} \geq (1+\epsilon)mp] \leq \exp\left(-\frac{\epsilon^2 mp}{3}\right).$$

Taking opposite probabilities, we have

$$\Pr[\widetilde{m} \leq (1+\epsilon)mp] \geq 1 - \exp\left(-\frac{\epsilon^2 mp}{3}\right).$$

We want ϵ to be very low here, so we are close to $\widetilde{m} \leq mp$, but it cannot be 0. Otherwise, the bound would become 0 — hardly useful. In order to have $(1+\epsilon)mp \leq m$

at the least, we need to set $\epsilon \leq (1-p)/p$. Of course, we want to set ϵ much lower than $(1-p)/p$, so we have something much lower than m. To make it lower, we let $\epsilon = (1-p)/(\ell p)$ where ℓ is some constant. In this setting, we begin with

$$\Pr\left[\widetilde{m} \leq \left(1 + \frac{1-p}{\ell p}\right) mp\right] \geq 1 - \exp\left(-\frac{(1+(1-p)/(\ell p))^2 mp}{3}\right).$$

This is difficult to work with in both places: the bound on $\widetilde{m}$ and the bound on the probability. We will loosen both of those bounds to simplify our expressions. If we let $\ell = m$, then we can loosen the bound on $\widetilde{m}$ as follows:

$$\begin{aligned}
\left(1 + \frac{1-p}{\ell p}\right) mp &= \left(\frac{(\ell-1)p+1}{\ell}\right) m \\
&= \left(\frac{(\ell-1)p}{\ell} + \frac{1}{\ell}\right) m \\
&\leq \left(p + \frac{1}{\ell}\right) m \\
&= mp + \frac{m}{\ell} \\
&= mp + \frac{m}{m} \\
&= mp + 1.
\end{aligned}$$

We will not explicitly loosen the bound on the probability. Instead we notice that m is in the numerator of the exponent. Thus for sufficiently large m, the bound is at least $1/2$. Finally, we conclude that $\widetilde{G}$ is $(mp+1)$-sparse for $p \geq 2/3$ and sufficiently large m. ■

It is not difficult to see that BASIC-RANDOM-SAMPLE performs very poorly since the approximation is within a factor of 5, p is not allowed to be very low, and we need m to be large. Using popular parlance, we would say this spectral sparsification algorithm runs in $O(m)$ time and produces $O(m)$ sized sparsifiers. The goal within the mathematical community is to construct spectral sparsification algorithms running in $O(m)$ time that create $O(n)$ sized sparsifiers. A running time of $O(m)$ is the best we can do since that amounts to looking at each edge once, and sparsifier of size $O(n)$ are the best we can do since that amounts to having nearly as

many edges as a tree. This algorithm's saving grace might be its $O(m)$ time, but its approximation is too poor. In the next section, we consider why this may be the case.

2.2 Choosing Edges

In the last section, we learned that sampling edges with fixed probability will produces poor guarantees. In this section, we explore what manner of choosing edges does work. The issues with the naive algorithm becomes apparent if we consider corner case graphs such as the barbell graph shown in Figure 2.1. The barbell graph is a graph with $2n$ vertices generated by combining two K_n graphs and joining them with a single edge.

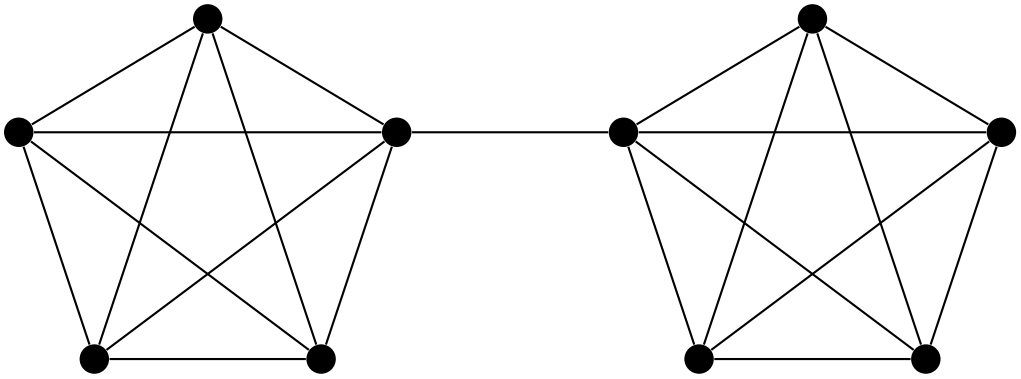


Figure 2.1: The barbell graph with $n = 5$

If we let G be a barbell graph, and if we gave it to `BASIC-RANDOM-SAMPLE` with probability p, then $\widetilde{G}$ will be disconnected with probability p. It is pertinent that a spectral sparsification algorithm does not disconnect the graph. Not only is this intuitively a property we want any algorithm to maintain, but disconnecting the graph is an easy way for $\widetilde{G}$ to not be a spectral approximation of G. This is because the number of connected components in a graph is given by the multiplicity of the Laplacian eigenvalue 0. For connected G, the multiplicity is 1, but if $\widetilde{G}$ is disconnected, then the multiplicity becomes 2, which means one of the eigenvalues is not guaranteed to be approximated. Therefore, we must choose edges such that we avoid disconnecting the graph. Intuitively, we can think of this as choosing the important edges of the graph.

This is exactly why a probabilistic approach is natural for spectral sparsification: we can assign probabilities in proportion to the importance of the edge.

The task now is to determine a property that represents how important an edge is to the connectivity of a graph. From the discussion on cuts given in Section 1.5, it would seem that such edges lie on the boundary of cuts with low isoperimetric number. Indeed, this intuition can produce nice spectral sparsifiers. It was the basis of the first spectral sparsification algorithm introduce by Spielman and Teng in their seminal paper [24]. In this thesis, we will consider effective resistances to assign importance to an edge. Similar to conductance, we can use effective resistances to target edges appearing in bottlenecks.

2.3 Spectral Sparsification Using Effective Resistances

Effective resistance can be roughly described as the overall resistance given by an edge if we treat the graph as a simple electrical network. They were used in the second major spectral sparsification algorithm [23]. We consider it, though, because this line of attack is simpler, has clear physical and combinatorial intuitions, and demonstrates how we can analyze the Laplacian by looking at its related matrices.

2.3.1 Modeling Resistance Networks

Mathematics and physics have often influenced the development of each other. The case is no different within graph theory. The idea here is to make a weighted graph model a specific type of electrical network called a resistance network. Such networks only contain resistors and sources of electric potentials. We will view the vertices as the sources of electric potentials and the edges as wires, which exhibit different resistances via the weights. To keep things simple, let an edge with weight 1 have resistance 1. Since edges model wires, the lack of an edge would be analogous to having no wire, so the electrons would never flow. This should represent an infinite resistance, and resistances should increase as weights decrease. Thus if e is an edge then

$$r_e = \frac{1}{w_e}$$

where r_e is the resistance of the edge, and w_e is its weight. Now that we know how components of a graph map onto a physical resistance network, we have to ensure the graph behaves like an actual circuit. To do this, we encode two electrical laws as constraints on the model.

Once the laws are in place, we will inject negative current into a vertex s and extract, the same valued but positive, current through a vertex t. We can think of this as having a battery with two wires, one attached to the positive side of the battery and another attached to the negative side. We inject current into the graph by attaching the negative end wire to s and the positive end wire to t to complete the circuit.

Our model will also assign electric potentials to vertices and currents to edges. The goal is to find assignments of electric potentials and currents that satisfy the electrical laws. We will learn that we can derive appropriate assignments for any value of current we inject. Once this ground work is laid, we will have a clear definition of effective resistances. Then we explore why we set sampling probabilities proportional to it.

The two laws, we are going to satisfy are Ohm's law and Kirchhoff's current law. Ohm's law tells us that

$$V = IR \implies I = \frac{V}{R}$$

where V is the voltage between two nodes, I is the amount (with direction) of current flow, and R is the resistance of the wire. If we let p be a vector of size n denoting energy potentials at each vertex, then Ohm's law tells us

$$i_{a,b} = \frac{p_a - p_b}{r_{a,b}} = w_{a,b}(p_a - p_b)$$

where $i_{a,b}$ denotes the current from a to b. Note that this naturally reflects the convention that current flows from high electric potential to low. That is if $p_a > p_b$, then the current will be positive, otherwise it will be negative. Furthermore

$$i_{a,b} = -i_{b,a}.$$

Let i be the vector of size m containing all the currents flowing across edges. Since i only contains a component for each edge once, we need to choose between using the current over (a, b) or (b, a). We will fix i to have positive values, so whenever $i_{a,b}$ is positive, we use (a, b). In addition, we call a the head and b the tail. If $i_{a,b}$ is negative, we use (b, a), and call b the head and a the tail. Recall how the Laplacian can be written as $L = M^T W M$ where

$$M((a,b),c) = \begin{cases} 1 & \text{if } c = a \\ -1 & \text{if } c = b \\ 0 & \text{otherwise} \end{cases}$$

and W is a diagonal $m \times m$ matrix with $W(e, e) = w_e$.

Surprisingly, we can use the oriented incidence matrix to construct a linear algebraic representation of Ohm's law. The rows of M have a single 1 and a single -1 wherever the head and tail are respectively, so we can write row (a, b) of M as $\chi_a - \chi_b$ which are the characteristic vectors of a and b respectively. It follows that

$$WMp = W\left[(\chi_a - \chi_b)^T p\right]$$

where $Mp = \left[(\chi_a - \chi_b)^T p\right]$ is a vector with m components indexed by the edges of G. The value of component (a, b) is equal to $(\chi_a - \chi_b)^T p = p_a - p_b$, so

$$WMp = W\left[p_a - p_b\right].$$

Since W is diagonal, we simply multiply each row of the vector by the corresponding diagonal entry. This allows us to write Ohm's law as

$$WMp = \left[w_{a,b}(p_a - p_b)\right] = \left[i_{a,b}\right] = i.$$

Kirchhoff's current law states: if a is a vertex and $(a, u_1), \cdots, (a, u_k)$ are its incident edges where a is a head, and $(u_{k+1}, a), \cdots (u_{d_a}, a)$ are the edges where a is a tail, then the sum of currents is zero, i.e.

$$i_{a,u_1} + \cdots + i_{a,u_k} + i_{a,u_{k+1}} + \cdots + i_{a,u_{d_a}} = \sum_{u \in N(a)} i_{a,u} = 0 \tag{2.5}$$

where $N(a)$ is the set of neighbors of a. Note that for our purposes $a \neq s, t$. For s and t, the law still holds, the $-k$ units of current injected into s leave s within its edges, but that initial injection is not portrayed by an edge. A similar situation happens with t. Thus when we sum of currents of edges incident to s and t, they will be k and $-k$ respectively since s is always a head and t is always a tail.

The form of Kirchhoff's law in (2.5) has positive terms and negative terms. Since we limited our representation of currents to positive terms, we can rewrite Kirchhoff's law as

$$\sum_{\substack{b \text{ s.t.} \\ ab \in E}} i_{a,b} - \sum_{\substack{b \text{ s.t.} \\ ba \in E}} i_{b,a} = 0$$

for $a \neq s, t$. Note the first sum is for when a is a head (a to b has positive current) and the second for when a is a tail (b to a has positive current). If we consider s and t as well, then we get, for arbitrary vertex a, that

$$\sum_{\substack{b \text{ s.t.} \\ ab \in E}} i_{a,b} - \sum_{\substack{b \text{ s.t.} \\ ba \in E}} i_{b,a} = \begin{cases} k & \text{if } a = s \\ -k & \text{if } a = t \\ 0 & \text{otherwise} \end{cases}.$$

Let i_{ext} be a vector with n components representing the external current flowing to each vertex. That is, we have k in coordinate s and $-k$ in coordinate t and 0 everywhere else like above since only s and t interact with external current. Thus we write our slightly modified version of Kirchhoff's law with

$$M^T i = i_{ext}.$$

To summarize, Ohm's law and Kirchhoff's law give us the following constraints:

$$i = WMv$$
$$M^T i = i_{ext}.$$

Putting these together, we get

$$M^T i = M^T W M p = Lp = i_{ext}$$

We have combined both constraints into one: $Lp = i_{ext}$. Given a graph and some choice of $-k$ current to inject, we can derive satisfying electric potentials at each vertex by solving for v. Once we have the potentials, we can use Ohm's law to solve for the currents and voltages from there. Recall the pseudo-inverse L^+ that we discussed in Subsection 1.3.5. We can use that to solve for p since $i_{ext} \perp \mathbf{1}$:

$$p = L^+ i_{ext}.$$

It is important to note that this solution is not unique. There are other solutions that can satisfy both laws as well. This will become important later when we make use of this fact.

The experienced reader may ask why Kirchhoff's voltage law is missing from this discussion. In fact, it is not. It is just hidden away in the other two laws. The voltage law states that the sum of voltages on edges contained in a cycle must be zero. Consider some cycle $a, b_1, \cdots, b_k, a$. The sum of voltages is given by

$$(p_a - p_{b_1}) + \cdots + (p_{b_k} - p_a) = (p_a - p_a) + (p_{b_1} - p_{b_1}) + \cdots + (p_{b_k} - p_{b_k}) = 0$$

In our model, we inject current into one vertex and extract it out of another. We use Ohm's law and Kirchhoff's current law to solve for the potential differences along each edge. The above equality shows that the sum of potential differences must be equal to zero on any cycle. If one wants to find the actual potential at the vertices, we can arbitrarily pick one, and solve for the others. This is demonstrated in the proof for Theorem 2.8.

2.3.2 Effective Resistances

In our model, we are allowed to inject and extract currents into single vertices such that the amount injected is equal to the amount extracted. The **effective resistance** between vertices a and b is defined as the voltage between a and b when -1 unit of current is injected at a and extracted from b. We can write this as

$$\mathcal{R}_{a,b} = p_a - p_b = (\chi_a - \chi_b)^T p = (\chi_a - \chi_b)^T L^+ i_{ext} = (\chi_a - \chi_b)^T L^+ (\chi_a - \chi_b).$$

We noticed before that $(\chi_a - \chi_b)$ is row (a, b) of M, so we can further write

$$\mathcal{R}_{a,b} = (\chi_a - \chi_b)^T L^+ (\chi_a - \chi_b) = M_{a,b} L^+ M_{a,b}^T$$

where $M_{a,b}$ is a row vector given by row (a, b) of M. In short, we can represent all effective resistances of G in a matrix calculated by $\mathcal{R} = ML^+M^T$. This is an $m \times m$ matrix indexed by edges on both sides with $\mathcal{R}((a, b), (a, b))$ holding the effective resistance for (a, b). We abuse notation, and let $\mathcal{R}_{a,b}$ denote the effective resistance as well.

It may seem odd that we call $p_a - p_b$ an "effective resistance" when it is actually a voltage. The name comes from the fact that $p_a - p_b$ is the resistance of the edge if we treated the entire network as an edge between a and b. Suppose we remove all edges but (a, b). By Kirchhoff's law, each edge we remove only causes the current moving through (a, b) to increase. When we are left with only (a, b), its current is 1 since we injected -1 unit of current into a. By Ohm's law, we have

$$1 = \frac{p_a - p_b}{r_{a,b}}$$

which means $r_{a,b} = p_a - p_b$. Thus the resistance of the edge when the entire network is treated as an edge is effectively $p_a - p_b$. Furthermore, injecting any other amount of current leaves the resistance unchanged. If we look at these two laws in a slightly different way, it becomes more clear why one would sparsify according to effective resistances.

Suppose we inject -1 units of current into a. This means a will always be the head in each of its edges. If $(a, b_1), \cdots, (a, b_{d_a})$ are the edges incident to a, then Kirchhoff's law states

$$i_{a,b_1} + \cdots + i_{a,b_{d_a}} = k.$$

Ohm's law allows us to rewrite this as

$$w_{a,b_1}(p_a - p_{b_1}) + \cdots + w_{a,b_{d_a}}(p_a - p_{b_{d_a}}) = k.$$

Now let us assume a has small degree. This essentially means there exists a bottleneck on the flow of current between a and the rest of the graph. With smaller degree, there

are fewer edges, which means each of the voltage terms [2] in the sum should be higher on average. This is why we bother considering the resistance model of a graph. It allows us to look at the flow of electrons, and use voltage calculations to see exactly where the bottlenecks are. There are many different voltages that can be induced depending on where we inject and extract current. The effective resistance simply provides a natural one that takes the connectivity of the graph into account.

2.3.3 Combinatorial Intuition for Effective Resistances

There is a rich connection between the study of electrical network models and random walks. We will show in particular that the effective resistance of an edge is exactly equal to the probability that the edge appears in a uniformly random spanning tree. A **spanning tree** is a spanning subgraph that is also a tree. Suppose our graph has a bottleneck, which corresponds to the boundary edges of some cut. This cut separates the vertices in the two parts. Any spanning tree must must have an edge crossing this bottleneck, otherwise it would not contain every vertex. Since there are relatively fewfew edges in a bottleneck, each edge has a relatively high probability of appearing in a random spanning tree. This is the main reason we decide to sparsify with respect to effective resistances. The higher the effective resistance, the more likely that edge appears in random spanning tree, which further means the more likely it appears in a bottleneck. Such edges are more important to the connectivity of the graph and must be sampled with higher probability.

A direct link between effective resistances and random spanning trees does not exist though. In order to make the link, we will take advantage of the connection between random walks and random spanning trees, and we will take advantage of the connection between effective resistances and random walks. To keep our analysis simple, we will limit ourselves, to the unweighted case. The connection between random walks and random spanning trees is given by the Aldous-Broder algorithm for

[2] Note that only one of the voltage terms will be the effective resistance, depending on where we take the current out.

generating random spanning trees [1, 7].

Algorithm 2 Aldous-Broder Spanning Tree Generator

Require: $G = (V, E)$ is a connected, simple graph
Require: $s \in V$
  function ALDOUS-BRODER-GEN(G, s)
    $E' \leftarrow \emptyset$
    $V' \leftarrow \emptyset$
    $T \leftarrow (V', E')$
    $currVertex \leftarrow s$
    $prevVertex \leftarrow null$
    while $V' \neq V$ **do**
      $nextVertex \leftarrow chooseAtRandom(neighbors(currVertex))$
      $prevVertex \leftarrow currVertex$
      $currVertex \leftarrow nextVertex$
      if $currVertex \notin V'$ **then**
        $V' \leftarrow V' \cup currVertex$
        $E' \leftarrow E' \cup (prevVertex, currVertex)$
        $T \leftarrow (V', E')$
      end if
    end while
    return T
  end function

The Aldous-Broder algorithm takes a graph G and starts a random walk at s. The function *chooseAtRandom*() is assumed to choose an element from the given set with uniform probability. The algorithm walks from vertex to vertex at random, i.e. if we are at vertex u, then we move to one of its neighbors (gotten from *neighbors*()) with probability $1/d_u$. As it walks, we build a spanning tree. Every time we walk to a vertex we have not seen before, we add the edge we traversed to get there into the spanning tree. We are done building the spanning tree once we have seen every vertex. It is a fact that this algorithm produces not just a random spanning tree, but a uniformly random spanning tree.

Theorem 2.6. *If G is an unweighted graph, s is some starting vertex in G, and T is the result of* ALDOUS-BRODER-GEN(G, s)*, then T is a spanning tree chosen uniformly at random from the set of spanning trees on G.*

Since we are adding edges to the spanning tree when we first see a vertex, an edge (s,t) appears in T if and only if a random walk reaches t for the first time via the edge (s,t). Therefore, the probability an edge (s,t) appears in a uniform random spanning tree is equal to the probability that a random walk reaches t for the first time via the edge (s,t). The connection between random walks and effective resistances tells us that this probability is given exactly by the effective resistance.

Theorem 2.7. *If G is an unweighted graph, then the probability that a random walk started at s reaches t via the edge (s,t) is equal to $\mathcal{R}_{s,t}$.*

Proof. Let C be the event that a random walk started at s reaches t via the edge (s,t). We want to find $\Pr[C]$. There are two ways for C to happen. In the first case C_1, we simply traverse (s,t) on the first step. In the other case C_2, we walk to some other neighbor of s and randomly walk until we see s again and traverse (s,t). Since these two cases are independent from each other

$$\Pr[C] = \Pr[C_1] + \Pr[C_2].$$

The probability of walking to any neighbor of s is $1/d_s$, so $\Pr[C_1] = 1/d_s$. Finding the probability of C_2 is harder. There are three steps to obtaining C_2. First we need to walk from s to one of its neighbors v where $v \neq t$. Then we need to randomly walk but reach s before ever reaching t. Because we are at s again, the third step is to simply start a new random walk and see if we reach t via (s,t). The first step happens with probability $1/d_s$ as well, and let $q_{s,t}(v)$ be the probability that a random walk starting at v reaches s before t. The third step, we know happens with $\Pr[C]$. Breaking it down like this, we can write

$$\Pr[C_2] = \sum_{v \in N(s)} \frac{1}{d_s} q_{s,t}(v) \Pr[C]$$

Combining this with C_1, we can recursively write

$$\Pr[C] = \Pr[C_1] + \Pr[C_2] \tag{2.6}$$

$$= \frac{1}{d_s} + \sum_{v \in N(s)} \frac{\Pr[C]}{d_s} + q_{s,t}(v). \tag{2.7}$$

By Lemma 2.8, we know $q_{s,t}(v) = p_v$ in the setting described in the statement of the lemma. With this new information, we can write (2.6) as

$$\Pr[C] = \frac{1}{d_s} + \sum_{v \in N(s)} \frac{\Pr[C]}{d_s} + p_v.$$

Solving for $\Pr[C]$, it follows that

$$\begin{aligned}
\Pr[C] &= \frac{1/d_s}{1 - \sum_{v \in N(s)} p_v/d_s} \\
&= \frac{1}{d_s - \sum_{v \in N(s)} p_v} \\
&= \frac{1}{\sum_{v \in N(s)} 1 - \sum_{v \in N(s)} p_v} \\
&= \frac{1}{\sum_{v \in N(s)} 1 - p_v} \\
&= \frac{1}{\sum_{v \in N(s)} p_s - p_v}.
\end{aligned}$$

The last equality is true because in the setting of Lemma 2.8, $p_s = 1$. By Ohm's law and the fact G is unweighted, we can write the denominator in terms of currents:

$$\Pr[C] = \frac{1}{\sum_{v \in N(s)} i_{s,v}}.$$

In the setting of the lemma, we also know that we have injected s with a $-1/\mathcal{R}_{s,t}$ unit current. By Kirchhoff's law, the input current (that was injected) and the output current (going out of incident edges) must be equal. Thus the denominator is equal to $1/\mathcal{R}_{s,t}$. Finally, we can write

$$\Pr[C] = \frac{1}{1/\mathcal{R}_{s,t}} = \mathcal{R}_{s,t}$$

as desired. ■

Lemma 2.8. *Let G is an unweighted graph, and define $q_{s,t}(v)$ to be equal to the probability that a random walk started at v reaches s before t. Finally, let p_v be the unique*

energy potential at v induced by treating G as a resistance network, injecting $-1/\mathcal{R}_{s,t}$ units of current at s, extracting the same amount from t, and constraining p with $p_s = 1, p_t = 0$. In this setting $q_{s,t}(v) = p_v$.

Proof. To show that $q_{s,t}(v) = p_v$, we will construct recursive formulations of both quantities, and show that their formulations are identical as well as their base cases. This will demonstrate that they must be equal since they are calculated in the same fashion.

Recall that Kirchhoff's current law states

$$\sum_{u \in N(v)} i_{v,u} = 0.$$

Ohm's law tells us that each term $i_{v,u}$ is given by $w_{v,u}(p_v - p_u)$. Since G is unweighted, we can drop the weight and get

$$\sum_{u \in N(v)} p_v - p_u = 0 \implies \sum_{u \in N(v)} p_v = \sum_{u \in N(v)} p_u.$$

Note that the left hand sums up p_v exactly d_v times, so

$$\begin{aligned} d_v p_v &= \sum_{u \in N(v)} p_u \\ \implies p_v &= \frac{1}{d_v} \sum_{u \in N(v)} p_u. \end{aligned} \tag{2.8}$$

This contains our recursive formulation of p_v, which can be used regardless of what amount of current is injected. To get the base cases, we perform a little trick by injecting a certain amount of current. Recall that we find $\mathcal{R}_{s,t}$ by setting i_{ext} to have 1 and -1 in s and t respectively. Let p^{eff} be the potential vector carrying p_s and p_t obtaining $\mathcal{R}_{s,t}$. We calculate p^{eff} with $p^{\text{eff}} = L^+ i$, and then define $\mathcal{R}_{s,t} = p_s^{\text{eff}} - p_t^{\text{eff}}$.

Suppose we injected $-1/\mathcal{R}_{s,t}$ instead. It follows that

$$\begin{aligned} p_s - p_t &= \left(L^+ i_{ext} \frac{1}{\mathcal{R}_{s,t}}\right)_s - \left(L^+ i_{ext} \frac{1}{\mathcal{R}_{s,t}}\right)_t \\ &= \frac{1}{\mathcal{R}_{s,t}}(p_s^{\text{eff}} - p_t^{\text{eff}}) \\ &= \frac{1}{\mathcal{R}_{s,t}}\mathcal{R}_{s,t} \\ &= 1. \end{aligned}$$

By injecting $-1/\mathcal{R}_{s,t}$, we have the constraint $p_s - p_t = 1$. Suppose we add $p_s = 1$ and $p_t = 0$ as our base cases. These can be thought of as additional constraints on p. This may cause the p we have derived to no longer satisfy $Lp = (1/\mathcal{R}_{s,t})i_{ext}$, which encodes both laws. Therefore we need to show there exists p' satisfying our constraints such that $Lp' = i_{ext}$. Thankfully, this amounts to only checking if there exists a solution satisfying the system of equations generated by (2.8). There are $n-2$ unknown potentials, and (2.8) gives an equation to find each one, so there are $n-2$ equations in the system as well, so there exists a unique solution p'.

To summarize, if we inject $-1/\mathcal{R}_{s,t}$ units of current into s, then there exists a potentials vector p' such that

$$\begin{aligned} p'_s &= 1 \\ p'_t &= 0 \\ p'_v &= \frac{1}{d_v} \sum_{u \in N(v)} p'_u. \end{aligned} \tag{2.9}$$

Now we need to show that we can get an identical recursive formulation of $q_{s,t}(v)$. Since $q_{s,t}(v)$ represents the probability that a random walk started at v reaches s before t, it is clear that $q_{s,t}(s) = 1$ since it starts at s, and $q_{s,t}(t) = 0$ since it already reached t before any other vertex. For $v \neq s, t$, we can derive a recursive definition for $q_{s,t}(v)$.

Let us start a walk on v. Any walk that reaches s before t must make a first move to one of the neighbors of v. Suppose v has k neighbors, and let $q^i_{s,t}(v)$ be

the probability of starting a random walk at v, moving to the ith neighbor of v, and reaching s before t. Due to mutual exclusiveness, we can write

$$q_{s,t}(v) = \sum_{i=1}^{k} q_{s,t}^{i}(v).$$

Now consider any $q_{s,t}^{i}(v)$. Making the transition from v to its ith neighbor happens with probability $1/d_v$. We will call this neighbor u_i. Once we are at u_i, we want to randomly walk and reach s before t. Notice that continuing a random walk is the same as starting a new one, and the probability of starting at u_i and seeing s before t is given by $q_{s,t}(u)$. Therefore

$$q_{s,t}(v) = \sum_{u \in N(v)} \frac{1}{d_v} q_{s,t}(u) = \frac{1}{d_v} \sum_{u \in N(v)} q_{s,t}(u).$$

Since we have the base cases $q_{s,t}(s) = 1$ and $q_{s,t}(v) = 0$, the recursive formulation is identical to the one given in (2.9), so they are the same quantity. ■

2.3.4 First Try at Sparsifying with Effective Resistances

In Section 2.1, we noticed that sampling edges would be an appropriate means of sparsifying a graph since we expect to only choose a subset of the original edges. In Section 2.2, we learned that we have to be careful about choosing edges. In particular, we learned that we should choose edges with probability proportional to how "important" they are to the overall connectivity of the graph. In the previous two subsections, we demonstrated how effective resistances have hidden connections to random walks and spanning trees, which make it an excellent choice for setting our edge probabilities.

In this subsection, we put this knowledge to use and make a modification to BASIC-RANDOM-SAMPLE that sets edge probabilities equal to the maximum of $1/2$ and the effective resistance for that edge. We put a lower bound on the probabilities, so that the Chernoff bound does not produce unwieldy results. The algorithm in full detail is given below in Algorithm 3.

In this algorithm, *erList* represents a list, indexed by edges, of effective resistances calculated by *resistances*(). To calculate the effective resistances: first calculate

Algorithm 3 Simple Sparsification by Effective Resistances

Require: $G = (V, E)$ is a connected, simple graph
  function SIMPLE-SPARSIFY-BY-ER(G, ϵ)
    $\widetilde{E} \leftarrow \emptyset$
    $\widetilde{G} \leftarrow (V, \widetilde{E})$
    $eRList \leftarrow resistances(G)$
    $m \leftarrow |E|$
    for all $(u, v) \in E$ **do**
      $edgeProb \leftarrow \max(eRList[(u, v)], 1/2)$
      $includeEdge? \leftarrow true$ with probability $edgeProb$
      if $includeEdge?$ **then**
        $\widetilde{G} \leftarrow (V, \widetilde{E} \cup (u, v, 1/edgeProb)$
      end if
    end for
     return $\widetilde{G}$
  end function

$L+$ using any algorithm that finds eigenvalues and eigenvectors, and then compute ML^+M^T through usual matrix multiplication. This can all be done in polynomial time.

We will apply the same sort of analysis as in Theorem 2.5, but with an additional layer of complexity to garner better results. First, we will demonstrate that $\mathcal{R}$ is a projection matrix and prove some other useful properties. By showing $\mathcal{R}$ is a projection matrix, we can represent it with $\mathcal{R}\mathcal{R}$. Additionally, it allows us to map arbitrary $x \in \mathbb{R}^n$ onto some y in the image of $\mathcal{R}$. We will then apply Chernoff Bounds to $(y^T\mathcal{R}W\mathcal{R}y)/(y^T\mathcal{R}\mathcal{R}y)$ where W contains the new weights of $\widetilde{G}$. We then work backwards from our mapping to get back $x^T\widetilde{L}x$ and x^TLx. Finally, we will demonstrate how our Chernoff bound results in $\mathcal{R}$ transfer over to L.

We begin by letting p_e equal the sampling probability assigned to edge e. That is

$$p_e = \max\left(\frac{(2+\sqrt{\sqrt{2}\ln n})}{\ln(n)}, \mathcal{R}(e,e)\right)$$

according to our algorithm. We begin by again noting that $L = MM^T$. Furthermore, $\widetilde{L} = M^TWM$ where W is a $m \times m$ matrix indexed by edges, where diagonal elements

are given by

$$M_{e,e} = \begin{cases} \frac{1}{p_e} & \text{with probability } p_e \\ 0 & \text{with probability } 1 - p_e \end{cases}.$$

All other elements in W are 0. It is easy to check that $\mathrm{E}[M] = I$, and thus $\mathrm{E}\left[\widetilde{L}\right] = L$, so we have maintained expectations as before. Now we give some useful properties of $\mathcal{R}$.

Lemma 2.9. *If $\mathcal{R} = ML^+M^T$ is the matrix containing effective resistances of a graph G on its diagonal elements, then:*

1. *$\mathcal{R}$ is a projection matrix*
2. *$\mathrm{im}(\mathcal{R}) = \mathrm{im}(M)$*
3. *The eigenvalues of $\mathcal{R}$ are 1 with multiplicity $n-1$ and 0 with multiplicity $m-n+1$.*
4. *$\mathcal{R}(e,e) = \|\mathcal{R}(\cdot, e)\|^2$ where $\mathcal{R}(\cdot, e)$ denotes column e of $\mathcal{R}$.*

Proof.

1. Observe that

$$\begin{aligned} \mathcal{R}^2 &= (ML^+M^T)(ML^+M^T) \\ &= ML^+(M^TWM)L^+M^T \\ &= ML^+LL^+M^T. \end{aligned}$$

   Recall that L^+L is a projection matrix, so it acts as an identity on its image, which is L^+. This means for all $x \in \mathrm{im}(L^+)$, it follows that $L^+Lx = x$. Each column of L^+ is in its image as well (because of the standard basis vectors), so

$$\begin{aligned} \mathcal{R}^2 &= ML^+LL^+M^T \\ &= W^{1/2}ML^+M^TW^{1/2} \\ &= \mathcal{R}. \end{aligned}$$

   By definition, this equality makes $\mathcal{R}$ a projection matrix.

2. Let $x \in \mathrm{im}(\mathcal{R})$. Then $\mathcal{R}y = x$ for some $y \in \mathbb{R}^m$. It follows

$$\begin{aligned} &\mathcal{R}y = x \\ \implies &ML^+M^Ty = x \\ \implies &(M^TM)L^+M^Ty = M^Tx \\ \implies &(M^TM)L^+M^Ty = M^Tx \\ \implies &LL^+M^Ty = M^Tx. \end{aligned}$$

If we let $z = L^+M^Ty$, it follows that $z \in \text{im}(L^+) = \text{im}(L^+L)$ due to M^Ty. Since L^+L acts as an identity on its image, we can write

$$\begin{aligned}
&\mathcal{R}y = x \\
&\implies Lz = M^Tx \\
&\implies ML^+Lz = ML^+M^Tx \\
&\implies Mz = ML^+M^Tx \\
&\implies Mz = \mathcal{R}x \\
&\implies Mz = x.
\end{aligned}$$

The last line tells us that $x \in \text{im}(M)$.

The other direction is easier to show. First note that $\ker(M) = \text{span}(\mathbf{1})$ since $M\mathbf{1} = 0$. Let $x \in \text{im}(M)$, then there exists some y such that $My = x$. Furthermore, we can choose $y \perp \mathbf{1}$, so $y \in \text{im}(L^+L)$.

$$\begin{aligned}
\mathcal{R}x &= ML^+M^Tx \\
&= ML^+M^TMy \\
&= ML^+Ly \\
&= My \\
&= x.
\end{aligned}$$

Thus we have $\text{im}(\mathcal{R}) = \text{im}(M)$ since $x \in \text{im}(\mathcal{R})$.

3. Since $\mathcal{R}$ is a projection matrix, so its eigenvalue are 0 and 1. We also know $\ker(M) = \ker(L)$ has 1 dimension. Furthermore, $\text{im}(M)$ has $n-1$ dimensions. Because $\text{im}(\mathcal{R}) = \text{im}(M)$, $\text{im}(\mathcal{R})$ has $n-1$ dimension as well. Since the non-zero eigenvalues are produced by eigenvectors in $\text{im}(\mathcal{R})$, the eigenvalue 1 has multiplicity $n-1$. The remaining $m-n+1$ eigenvalues must then be 0.

4. Since $\mathcal{R} = \mathcal{R}^2$, and $\mathcal{R}$ is symmetric, it follows

$$\mathcal{R}(e,e) = (\mathcal{R}\mathcal{R})(e,e) = (\mathcal{R}(\cdot,e))^T\mathcal{R}(\cdot,e) = \|\mathcal{R}(\cdot,e)\|^2.$$

■

Now consider the matrix $\mathcal{R}W\mathcal{R}$. This is essentially the effective resistances matrix $\mathcal{R}$ scaled by the new weights. We consider it because we can relate it directly to $\widetilde{L}$. Let $x \in \mathbb{R}^n$, and $y = Mx$. Since $\widetilde{L} = M^TWM$, we can write

$$x^T\widetilde{L}x = x^TM^TWMx = (Mx)^TW(Mx) = y^TWy.$$

We also know $\text{im}(\mathcal{R}) = \text{im}(M)$ and $\mathcal{R}$ is a projection matrix, so $x^T\widetilde{L}x = y^T\mathcal{R}W\mathcal{R}y$. Similarly, we can show $x^TLx = y^T\mathcal{R}\mathcal{R}y$. We will make use of the direct relationship between these two quadratic forms in order to show that $\widetilde{G}$ is a spectral approximation of G.

Theorem 2.10. *Let G be a connected, unweighted, and simple graph. If $\widetilde{G}$ is the output of* SIMPLE-SPARSIFY-BY-ER(G)*, then $\widetilde{G}$ is an $(0, 1+\sqrt{7})$-spectral approximation of G with probability at least $1/2$.*

Proof. Proving this theorem will be a fairly straight forward application of Chernoff bounds. The random variable we consider is

$$X = \frac{y^T\mathcal{R}W\mathcal{R}y}{y^T\mathcal{R}\mathcal{R}y} = \frac{y^TWy}{y^Ty}$$

where $y \in \text{im}(\mathcal{R})$. Notice that we can write $X = \sum_{e\in E} X_e$ where

$$X_e = \frac{y^TW_ey}{y^Ty}$$

and W_e is a $m \times m$ matrix indexed by edges where every element is 0, except for $W_e(e,e)$, which is equal to $W(e,e) = 1/p_e$. We will apply the Chernoff bound given in Theorem 2.4. First we need to compute the expected value and get an upper bound on the values of X_e. It follows that

$$\text{E}\left[X_e\right] = \text{E}\left[\frac{y^TW_ey}{y^Ty}\right] = p_e\left(\frac{y_e^2}{y^Typ_e}\right) + (1-p_e)(0) = \frac{y_e^2}{y^Ty}.$$

Thus we can easily see

$$\text{E}\left[X\right] = \sum_{e\in E}\text{E}\left[X_e\right] = \sum_{e\in E}\frac{y_e^2}{y^Ty} = \frac{y^Ty}{y^Ty} = 1.$$

To find the upper bound, we need to notice that W_e is a diagonal and symmetric matrix. Because it is symmetric, the maximum value of y^TW_ey is given by its largest eigenvalue. Because it is diagonal, the eigenvalues are given by the diagonal elements. Since there is only one non-zero diagonal element, namely $1/p_e$, it is clear that y^TW_ey takes values in $[0, 1/p_e]$. But we need an upper bound over all possible e. We can easily

do this by giving a lower bound for p_e. If p is this lower bound, and we apply Chernoff bounds, we would get a bound of

$$1 - \exp\left(-\frac{\epsilon^2 p}{(2+\epsilon)}\right).$$

We can simplify things by making sure $p = (2+\epsilon)/k$ for some $k \geq 2+\epsilon$. This changes our problem to finding nice values for ϵ and k. Note that the bound

$$1 - \exp\left(-\frac{\epsilon^2}{k}\right)$$

is just above $1/2$ if we have $\epsilon^2/k = 7/10$. Therefore, we can get an easy spectral approximation if we let $\epsilon = \sqrt{7}$ and $k = 10$. In this case $p = (2+\sqrt{7})/10 \leq 1/2$. This is why write in the algorithm that the edges are chosen with probability at least $1/2$.

After applying Theorem 2.4, we know

$$\Pr\left[\frac{y^T W y}{y^T y} \geq (1+\epsilon)\right] \leq \exp\left(-\frac{(2+\epsilon)\epsilon^2}{(2+\epsilon)\ln n}\right).$$

Multiplying through with $y^T y$ and simplifying gives us

$$\Pr\left[y^T W y \geq (1+\epsilon) y^T y\right] \leq \exp\left(-\frac{\epsilon^2}{\ln n}\right).$$

After taking the opposite probability, we have

$$\Pr\left[y^T W y \leq (1+\epsilon) y^T y\right] \geq 1 - \exp\left(-\frac{\epsilon^2}{k}\right) \tag{2.10}$$

Since we let $\epsilon = \sqrt{7}$ and $k = 10$, we have

$$\Pr\left[y^T W y \leq (1+\epsilon) y^T y\right] \geq 1 - \exp\left(-\frac{7}{10}\right) \geq 1/2 \tag{2.11}$$

If we choose arbitrary $x \in \mathbb{R}^n$, we can always find $y \in \mathrm{im}(\mathcal{R})$ such that $x^T \widetilde{L} x = y^T W y$ and $x^T L x = y^T y$. From here, we can apply (2.11) and work backwards to get

$$0 \leq x^T \widetilde{L} x \leq (1+\epsilon) x^T L x$$

We conclude, with probability at least $1/2$, that $\widetilde{G}$ is a $(0, 1+\sqrt{7})$-spectral approximation of G. ■

This shows some improvement over BASIC-RANDOM-SAMPLE. First, we removed a parameter, which makes it easier to use. Second, our probabilities of choosing edges are near $1/2$ rather than $2/3$, which means we expect to get a more sparse graph. Finally, $1 + \sqrt{7} \approx 3.65$ so we have a slightly better approximation as well. Nevertheless, it is still not a good enough approximation, especially on the lower end. We can do better — we only need to employ more powerful proof techniques.

2.3.5 A Better Algorithm for Sparsifying with Effective Resistances

The law of large numbers is a statistical theorem that essentially states the sample average will become closer to the population average as we take more samples. In terms of sparsifying graphs, we can think of the experiment as sampling a single edge from the graph. Since we ensure that $\mathrm{E}\left[x^T \widetilde{L} x\right] = x^T L x$, the law of large numbers should tell us that sampling a single edge many times will cause $x^T \widetilde{L} x$ to become closer to $x^T L x$. While Chernoff bounds can put bounds on the probability that we are far away from the expected value, the law of large numbers can make sure we are close to the expected value. In particular, we will make use of a theorem that is similar to the law of large numbers, except that it applies only to symmetric rank 1 matrices.

Because we are now sampling a single edge from the graph many times with replacement rather than iterating over the edge set and sampling each edge, our algorithm will have some modifications. The major change we need to make is our choice of probabilities. In SIMPLE-SPARSIFY-BY-ER we set probabilities equal to effective resistances (with some care to ensure probabilities did not get too low). This will not work here because we are sampling a single edge from the entire edge set. If we set the probability distribution over E to be equal to effective resistances, then the sum would not be 1, making it an invalid probability distribution. In order to account for the edge importance information provided by effective resistances while ensuring the probabilities sum to 1, we will simply set

$$p_e = \frac{\mathcal{R}(e, e)}{n - 1}.$$

Since we are not using Chernoff bounds in our analysis, we will not need to introduce a lower bound on the probabilities. Furthermore, the denominator of $n-1$ comes from the fact that the sum of effective resistances is equal to $n-1$. This is easy to see since

$$\sum_{e \in E} \mathcal{R}(e,e) = \mathrm{Tr}(\mathcal{R}) = n-1$$

since $\mathcal{R}$ has $n-1$ non-zero eigenvalues. This exhibits a connection with random spanning trees since any spanning tree has $n-1$ edges.

Algorithm 4 Sparsification by Effective Resistances with Repeated Sampling [23]

Require: $G = (V, E)$ is a connected, simple graph
Require: q is a positive integer denoting the number of times to sample edges
  function SPARSIFY-BY-ER(G, q)
    $\widetilde{E} \leftarrow \emptyset$
    $\widetilde{G} \leftarrow (V, \widetilde{E})$
    $eRList \leftarrow resistances(G)$
    $edgeProbs \leftarrow eRList/(n-1)$
    $m \leftarrow |E|$
    loop q times
      $e \leftarrow sample(E, edgeProbs)$
      $\widetilde{G} \leftarrow (V, \text{ADJUST-WEIGHT}(\widetilde{E}, e))$
    end loop
     return $\widetilde{G}$
  end function
  function ADJUST-WEIGHT($\widetilde{E}, (u, v)$)
    if $(u, v) \in \widetilde{E}$ **then**
      $w \leftarrow$ weight of edge $(u, v) \in \widetilde{E}$
      $\widetilde{E} \leftarrow \widetilde{E} - (u, v, w)$
      $(u, v, w') \leftarrow (u, v, w + 1/qp_e)$
      $\widetilde{E} \leftarrow \widetilde{E} \cup (u, v, w')$
    else
      $\widetilde{E} \leftarrow (u, v, 1/qp_e)$
    end if
  end function

Before, we reweighted edges with $1/p_e$ to ensure $\mathrm{E}[W] = I$, and thus $\mathrm{E}[x^T \widetilde{L} x] = x^T L x$. We did this assuming we only look at each edge once, however. Now that we

might look at edges multiple times, it is necessary to change our reweighting scheme. [3] Instead, we will change the edge weights as we sample them more times. In particular, we reweight edges with

$$W(e,e) = \frac{q_e}{qp_e}$$

where q_e is the number of time edge e was sampled out of the q total samplings. We can think of this as telling us how much of edge e is included into $\widetilde{G}$. It is easy to check that expectations are maintained with this reweighting scheme. The algorithm in full detail is given in Algorithm 4.

To prove guarantees on this theorem, we will not analyze the quadratic form directly. Instead, we will analyze the spectral norm of the difference between $\mathcal{R}W\mathcal{R}$ and $\mathcal{R}\mathcal{R}$. If we can show this spectral norm is small, then we have a spectral-approximation. This is due to the relationship between eigenvalues and the quadratic form as demonstrated by Courant-Fischer.

Definition 2.1: *Spectral Norm*
The spectral norm of a matrix M with eigenvalues $\lambda_1, \cdots, \lambda_n$ is equal to $\max_i|\lambda_i|$.

Lemma 2.11. *If $\|\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R}\| \leq \epsilon$ with $0 \leq \epsilon \leq 1$, then $\widetilde{G}$ is a $(1-\epsilon, 1+\epsilon)$-spectral approximation of G.*

Proof. Since the eigenvalues for symmetric matrices are given by the Rayleigh quotients, and $\mathcal{R}$ and W are symmetric, the assumption above for $\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R}$ is equivalently given by

$$\begin{aligned}
\|\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R}\| &= \sup_{y \neq 0 \in R^m} \frac{|y^T(\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R})y|}{y^T y} \\
&= \sup_{y \neq 0 \in R^m} \frac{|y^T\mathcal{R}(W - I)\mathcal{R}y|}{y^T y} \\
&\leq \epsilon.
\end{aligned}$$

[3] Applying the old reweighting scheme to the new problem will cause the computation of expected value to become troublesome due to the inclusion-exclusion principle.

If we restrict ourselves to $y \in \text{im}(\mathcal{R}) = \text{im}(M)$, then

$$\sup_{y \neq 0 \in \text{im}(M)} \frac{|y^T \mathcal{R}(S - I)\mathcal{R}y|}{y^T y} \leq \epsilon.$$

Since $y \in \text{im}(\mathcal{R})$ and $\mathcal{R}$ is a projection matrix, we know $\mathcal{R}y = y$. Furthermore, we can write that $y = Mx$ for some $x \in \mathbb{R}^n$ because $y \in \text{im}(M)$. Using these two facts it follows

$$\begin{aligned}
\|\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R}\| &= \sup_{y \neq 0 \in \text{im}(M)} \frac{|y^T \mathcal{R}(W - I)\mathcal{R}y|}{y^T y} \\
&= \sup_{y \neq 0 \in \text{im}(M)} \frac{|y^T (W - I)y|}{y^T y} \\
&= \sup_{x \in \mathbb{R}^n} \frac{|(Mx)^T (W - I)(Mx)|}{(Mx)^T (Mx)} \\
&= \sup_{x \in \mathbb{R}^n} \frac{|(x^T M^T)(W - I)(Mx)|}{x^T M^T M x} \\
&= \sup_{x \in \mathbb{R}^n} \frac{|x^T M^T W M x - x^T M^T I M x|}{x^T L x} \\
&= \sup_{x \in \mathbb{R}^n} \frac{|x^T \widetilde{L} x - x^T L x|}{x^T L x} \\
&= \sup_{x \in \mathbb{R}^n} \frac{|x^T (\widetilde{L} - L) x|}{x^T L x} \\
&\leq \epsilon.
\end{aligned}$$

The above line of reasoning tells us that the spectral norm of $\widetilde{L} - L$ is exactly equal to the spectral norm of $\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R}$. Because we assumed that the spectral norm in the latter is small, the results transfer over to the former. If we also note that, for any particular choice of x, it holds that

$$\frac{|x^T (\widetilde{L} - L) x|}{x^T L x} \leq \sup_{x \in \mathbb{R}^n} \frac{|x^T (\widetilde{L} - L) x|}{x^T L x}.$$

we can derive that

$$\begin{aligned} x^T\widetilde{L}x &= x^TLx + x^T\widetilde{L}x - x^TLx \\ &= x^TLx + x^T(\widetilde{L} - L)x \\ &= x^TLx\left(1 + \frac{x^T(\widetilde{L} - L)x}{x^TLx}\right) \\ &\leq x^TLx\left(1 + \frac{|x^T(\widetilde{L} - L)x|}{x^TLx}\right) \\ &\leq x^TLx(1+\epsilon). \end{aligned}$$

Similarly, we can show $(1-\epsilon)x^TLx \leq x^T\widetilde{L}x$. Thus $\widetilde{G}$ is a $(1-\epsilon, 1+\epsilon)$-spectral approximation of G as desired. ■

Now that we can use the spectral norm to show that we have a spectral approximation, we only need to show that the spectral norm truly is small. We make use of a law of large numbers theorem to show this. The theorem, given in Theorem 2.12, is complicated but can be broken down. We have some set of vectors S and we can sample them with different probabilities as described in probability distribution p. Suppose that we sampled q of them to get $y_1, \cdots, y_q$. Furthermore, suppose we can upper bound the lengths of all vectors in S by M and upper bound $\mathrm{E}[yy^T]$, which is the average outer product matrix definable by vectors in S. If this is all the case, then we can upper bound the expected difference between our average sample outer product matrix and the average outer product matrix over the entire population. In particular, as q increases, the upper bound becomes smaller, which means the difference between the average sample matrix and the average population matrix becomes smaller. Therein lies the connection to the law of large numbers.

Lemma 2.12. [22] *Let p be a probability distribution over a subset of $S \subseteq \mathbb{R}^m$ such that $\sup_{y\in S}\|y\| \leq M$, and $\|\mathrm{E}[yy^T]\| \leq 1$. If $y_1, \cdots, y_q$ are independent samples drawn from p, then*

$$\mathrm{E}\left[\left\|\frac{1}{q}\sum_{i=1}^{q} y_iy_i^T - \mathrm{E}[yy^T]\right\|\right] \leq \min\left(CM\sqrt{\frac{\log q}{q}}, 1\right)$$

where C is some constant.

With this lemma, we are now ready to show that we can produce high quality spectral sparsifiers.

Theorem 2.13. *Let G be a connected, unweighted, and simple graph. Furthermore, let $1/\sqrt{n} \leq \epsilon \leq 1$ and $q = 9C^2 n \log n(1/\epsilon^2)$. If $\widetilde{G}$ is the output of* SPARSIFY-BY-ER(G, q)*, then $\widetilde{G}$ is an $(1-\epsilon, 1+\epsilon)$-spectral approximation of G with probability at least $1/2$.*

Proof. We begin by noting that $\mathcal{R}W\mathcal{R} = \sum_{e \in E} W(e,e)\mathcal{R}(\cdot, e)\mathcal{R}(\cdot, e)^T$. Note that the outer product of columns of $\mathcal{R}$ create rank 1 matrices. We can then write

$$\begin{aligned}
\mathcal{R}W\mathcal{R} &= \sum_{e \in E} W(e,e)\mathcal{R}(\cdot, e)\mathcal{R}(\cdot, e)^T \\
&= \sum_{e \in E} \frac{q_e}{q p_e}\mathcal{R}(\cdot, e)\mathcal{R}(\cdot, e)^T \\
&= \frac{1}{q}\sum_{e \in E} q_e \frac{\mathcal{R}(\cdot, e)}{\sqrt{p_e}}\frac{\mathcal{R}(\cdot, e)^T}{\sqrt{p_e}}
\end{aligned}$$

If we let Y be a random variable where

$$Y = \frac{1}{\sqrt{p_e}}\mathcal{R}(\cdot, e)$$

with probability p_e, then we can further write

$$\begin{aligned}
\mathcal{R}W\mathcal{R} &= \frac{1}{q}\sum_{e \in E} q_e \frac{\mathcal{R}(\cdot, e)}{\sqrt{p_e}}\frac{\mathcal{R}(\cdot, e)^T}{\sqrt{p_e}} \\
&= \frac{1}{q}\sum_{i=1}^{q} y_i y_i^T
\end{aligned}$$

where $y_1, \cdots, y_q$ are sampled with replacement from Y independently. Essentially, instead of iterating over each edge and adding some $y \in Y$ exactly q_e times, we instead iterate over each sample and add the corresponding y_i to the sum. This will effectively produce the same sum.

If we let $p = Y$ and notice that Y takes values that are vectors in some subspace of $\mathbb{R}^m$, then we can see that we are almost ready to apply Theorem 2.12. It just remains to bound the lengths and spectral norm. To bound the norm, we notice

$$\begin{aligned}
\|y\| &= \left\| \frac{1}{\sqrt{p_e}} \mathcal{R}(\cdot, e) \right\| \\
&= \frac{1}{\sqrt{p_e}} \|\mathcal{R}(\cdot, e)\| \\
&= \frac{1}{\sqrt{p_e}} \sqrt{\mathcal{R}(\cdot, e)^T \mathcal{R}(\cdot, e)} \\
&= \frac{1}{\sqrt{p_e}} \sqrt{\mathcal{R}(e, e)} \\
&= \sqrt{\frac{n-1}{\mathcal{R}(e, e)}} \sqrt{\mathcal{R}(e, e)} \\
&= \sqrt{n-1}.
\end{aligned}$$

This means we have $M = \sqrt{n-1}$. To get an upper bound on the spectral norm, we first note that

$$\mathrm{E}\left[yy^T\right] = \sum_{e \in E} p_e \left(\frac{1}{\sqrt{p_e}} \right)^2 \mathcal{R}(\cdot, e)\mathcal{R}(\cdot, e)^T = \sum_{e \in E} \mathcal{R}(\cdot, e)\mathcal{R}(\cdot, e)^T = \sum_{e \in E} \mathcal{R}\mathcal{R} = \mathcal{R}.$$

Since the non-zero eigenvalues of $\mathcal{R}$ are 1, that means $\|\mathcal{R}\| = 1$. After applying Theorem 2.12, we learn

$$\mathrm{E}\left[\|\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R}\|\right] \leq \min\left(C\sqrt{n-1}\sqrt{\frac{\log q}{q}}, 1 \right). \tag{2.12}$$

For the sake of argument, let us assume we can bound the above expectations by δ. In order to apply Theorem 2.11, we need to show $\|\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R}\| \leq \epsilon$. But (2.12) only gets us the expected value of the norm. In order to get a bound on the norm itself, we make use of Markov's Inequality.

$$\Pr\left[\|\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R}\| \geq \epsilon\right] \leq \frac{\mathrm{E}\left[\|\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R}\|\right]}{\epsilon}.$$

If we take opposite probabilities, we get

$$\Pr\left[\|\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R}\| \leq \epsilon\right] \geq 1 - \frac{\mathrm{E}\left[\|\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R}\|\right]}{\epsilon}.$$

Furthermore, since we assumed we bounded the expectation above by δ, we can write

$$\Pr\left[\|\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R}\| \leq \epsilon\right] \geq 1 - \frac{\delta}{\epsilon}.$$

The goal now is to actually find a bound δ. which produces a nice value for the probability bound. In particular, we will show demonstrate that $\delta = \epsilon/2$ truly is an upper bound for the expectation. If we achieve this, then our probability bound becomes $1/2$. Therefore, we need choose q such that

$$C\sqrt{n-1}\sqrt{\frac{\log q}{q}} \leq \frac{\epsilon}{2}. \tag{2.13}$$

There are many choices of q that could work, but we want to keep the number of samples low to get bounds on the number of edges in $\widetilde{G}$. Therefore, we would want to find the minimum such q. Given a particular graph with n vertices and value for ϵ, we can find the minimum using optimization algorithms. A closed form solution, or at least a bound on it, will be harder to get, and is out of the scope of this thesis.

Instead, we can work backwards from (2.13), make some guesses for what q should be, check if those values of q work, make modifications, and repeat. After doing this, if we set $q = 9C^2 n \log n/\epsilon^2$ and enforce that $\epsilon \geq 1/\sqrt{n}$, then we find that

$$\begin{aligned}
\mathrm{E}\left[\|\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R}\|\right] &\leq \min\left(C\sqrt{n-1}\sqrt{\frac{\log q}{q}}, 1\right) \\
&\leq C\sqrt{n-1}\sqrt{\frac{\log(9C^2 n\log n/\epsilon^2)}{9C^2 n\log n/\epsilon^2}} \\
&\leq \epsilon\sqrt{\frac{C^2(n-1)\log(9C^2 n\log n/\epsilon^2)}{9C^2 n\log n}} \\
&\leq \epsilon\sqrt{\frac{(n-1)\log(9C^2 n\log n/\epsilon^2)}{9n\log n}}.
\end{aligned}$$

It remains to show that the term under the square root is less than or equal to $1/2$. If

we enforce that $\epsilon \geq 1/\sqrt{n}$, then

$$\begin{aligned}
\mathrm{E}\left[\|\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R}\|\right] &\leq \epsilon\sqrt{\frac{(n-1)\log(9C^2 n \log n/\epsilon^2)}{9n\log n}} \\
&\leq \epsilon\sqrt{\frac{(n-1)\log(9C^2 nn \log n)}{9n\log n}} \\
&= \epsilon\sqrt{\frac{(n-1)\log(9C^2 n^2 \log n)}{n\log n^9}} \\
&\leq \epsilon\sqrt{\frac{\log(9C^2 n^2 \log n)}{\log n^9}}.
\end{aligned}$$

It is easy to see that with sufficiently large n that the term under the square root will be less than or equal to $1/2$ since the denominator grows faster than the numerator. Thus we know

$$\mathrm{E}\left[\|\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R}\|\right] \leq \frac{\epsilon}{2}$$

and furthermore

$$\Pr\left[\|\mathcal{R}W\mathcal{R} - \mathcal{R}\mathcal{R}\| \leq \epsilon\right] \geq \frac{1}{2}$$

If we apply Theorem 2.11, then it follows that with probability at least $1/2$ that $\widetilde{G}$ is a $(1-\epsilon, 1+\epsilon)$-spectral approximation of G as desired. ■

Chapter 3

FURTHER READING

We have introduced spectral graph sparsification as a consequence of two motivations. In terms of applications, sparsification is a preprocessing step used to speed up computations on considerably large graphs. In terms of theory, sparsification can be seen as a generalization of the well-studied expander graphs (and cut-sparsifiers). Historically, Spielman and Teng were first motivated to build sparsifiers in order to construct approximate solvers for symmetric, diagonally-dominant systems of linear equations [24]. The first major spectral sparsification algorithm iterated over all edges and probabilistically chose whether to include them in the sparse output. However, the edge probabilities were set with respect to the conductance of the given graph. Conductance is a graph property that generalized the isoperimetric number to also take into account the degree sums of vertices in the two parts of a cut. The algorithm ran in $O(m \log^c m)$ time and generated sparsifiers with $O(n \log^c n)$ edges for some constant c. [1]. The second major algorithm is the one given by Spielman and Srivistava [23], which we explain in detail, that probabilistically sparsifies with respect to effective resistances. The authors further demonstrate that we can sparsify well with approximated effective resistances and with some other optimizations. In this environment, the algorithm ran in $O((m/\epsilon^2) \log^c(m/\epsilon^2))$ time and produced sparsifiers with $O((n \log n)\epsilon^2)$ edges. Both algorithms produce nearly linear sparsifiers in nearly linear time. The latter is simpler but slower depending on ones choice of ϵ.

Batson, Spielman, and Srivistava [4] prove the existence of linearly sized (in the edges) sparsifiers, and they provide a simple deterministic algorithm to compute

[1] All algorithms discussed here produce sparsifiers nearly spectrally identical to the given graph.

them. Their technique is similar to using effective resistances. Namely, they also reduce the problem to sparsifying $\mathcal{R}$, but they do not make use of repeated sampling and the law of large numbers. To prove the existence of these sparsifiers, they make of a proof technique named *the barrier method*, which also makes use of a physical model. In particular, it views eigenvalues as charged particles lying on a slope with barriers on either side of the eigenvalues. They essentially sparsify the identity matrix first in a deterministic way, and use $\mathcal{R}$ to transfer the results over the Laplacian. To sparsify the identity, they consider one of its decompositions and adjust each term in the decomposition such that the eigenvalues do not change too much. Controlling the eigenvalues is done by the "barrier functions." Their algorithm runs in $O(mn^3/\epsilon^2)$ time, which fails to beat the nearly linear time algorithms prior to it. That being said, it does produce linearly sized sparsifiers, obtaining the goal.

Koutis, Levin, and Peng [17] modify the algorithm for sparsifying by effective resistances to calculate approximate effective resistances in a faster way than in the original algorithm. Furthermore, they provide two other algorithms using looser approximations of effective resistances that run much faster. Their best algorithm runs in $O(m)$ time and produces $O(n \log n/\epsilon^2)$ assuming the input graph is dense. This meets the goal for the runtime, but also produces nearly linear sized sparsifiers. Allen-Zhu, Liao, and Orecchia [2] provide an advanced algorithm for constructing linearly sized sparsifiers proved to exist by Batson, Spielman, and Srivistava. The benefit of this algorithm is that it produces linearly sized sparsifiers, but it runs in nearly quadratic time. This is a few orders of magnitude of improvement over Batson, Spielman, and Srivistava's elementary algorithm.

Generalizing some of the techniques introduced by Allen-Zhu, Liao, and Orecchia [2], Lee and Sun [19] introduce an algorithm to generate $O(qn/\epsilon^2)$ sized sparsifiers, but in nearly linear time. Here, q is a variable proportional to the accuracy of the final sparsifier. They make use of effective resistances and the barrier method technique to prove their results, but they use a probabilistic variant of what Batson, Spielman, and Srivistava [4] did. The best algorithm to date was also introduced by Lee and Sun [18].

In a later paper, they improve on their previous result to produce an $O(n/\epsilon^2)$ sized sparsifier in $O((m/\epsilon^k)\log^c(m/\epsilon^k))$ time. Here k and c are some constants. In their paper, they consider the more general problem of sparsifying positive semi-definite matrices.

Generally speaking, the development of spectral sparsification started in the realm of spectral graph theory but moved into the realm of linear algebra as it became clear that tackling the more general problem was more fruitful. As such, there is a parallel transition from using graph-theoretic techniques like effective resistances to purely linear algebraic techniques. As for the current state of the problem: the latest algorithms are almost practically efficient. It remains for many of these to be implemented and examined empirially..

BIBLIOGRAPHY

[1] D. Aldous, *The random walk construction of uniform spanning trees and uniform labelled trees*, SIAM Journal on Discrete Mathematics (1990).

[2] Z. Allen Zhu, Z. Liao, and L. Orecchia, *Spectral sparsification and regret minimization beyond matrix multiplicative updates* (2015).

[3] N. Alon and V.D. Milman, λ_1 *isoperimetric inequalities for graphs, and superconcentrators*, Journal of Combinatorial Theory, Series B (1985).

[4] J. Batson, D. Spielman, and N. Srivastava, *Twice-ramanujan sparsifiers*, SIAM Journal on Computing (2012).

[5] A. Benczúr and D. Karger, *Approximating s-t minimum cuts in Õ(n2) time*, Proceedings of the twenty-eighth annual acm symposium on theory of computing, 1996.

[6] S. Boucheron, G. Lugosi, and O. Bousquet, *Concentration inequalities*, Springer Berlin Heidelberg, 2004.

[7] A. Broder, *Generating random spanning trees*, 30th annual symposium on foundations of computer science, 1989.

[8] A. Chakeri, H. Farhidzadeh, and L.O. Hall, *Spectral sparsification in spectral clustering*, 2016 23rd international conference on pattern recognition (icpr), 2016.

[9] P Chew, *There is a planar graph almost as good as the complete graph*, Proceedings of the second annual symposium on computational geometry, 1986.

[10] E.G. Coffman and R.L. Graham, *Optimal scheduling for two-processor systems*, Acta Informatica (1972).

[11] R. Cont and E. Tanimura, *Small-world graphs: Characterization and alternative constructions*, Advances in Applied Probability (2008).

[12] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to algorithms, third edition*, MIT Press, 2009.

[13] A. De Mauro, M. Greco, and M. Grimaldi, *A formal definition of big data based on its essential features*, Library Review (2016).

[14] G. Golub and C. Van Loan, *Matrix computations (3rd ed.)*, Johns Hopkins University Press, 1996.

[15] M Hilbert, *Quantifying the data deluge and the data drought*, SSRN Electronic Journal (2015).

[16] S. Hoory, N. Linial, and A. Wigderson, *Expander graphs and their applications*, Bulletin of the American Mathematical Society (2006).

[17] I. Koutis, A. Levin, and R. Peng, *Faster spectral sparsification and numerical algorithms for SDD matrices*, ACM Trans. Algorithms (2012).

[18] Y. Lee and H. Sun, *An sdp-based algorithm for linear-sized spectral sparsification*, Proceedings of the 49th annual acm sigact symposium on theory of computing, 2017.

[19] Y.T. Lee and H. Sun, *Constructing linear-sized spectral sparsification in almost-linear time*, 2015 ieee 56th annual symposium on foundations of computer science, 2015.

[20] B. Pavel, *A survey on pagerank computing*, Internet Mathematics (2005).

[21] F. Pietrucci and W. Andreoni, *Graph theory meets ab initio molecular dynamics: Atomic structures and transformations at the nanoscale*, Physical Review Letters (2011).

[22] M. Rudelson and R. Vershynin, *Sampling from large matrices: An approach through geometric functional analysis*, J. ACM (2007).

[23] D. Spielman and N. Srivastava, *Graph sparsification by effective resistances*, SIAM Journal of Computing (2011).

[24] D. Spielman and S Teng, *Spectral sparsification of graphs*, SIAM J. Comput. (2011).

[25] D. Spielman and S.H. Teng, *Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems*, SIAM Journal on Matrix Analysis and Applications (2014).

[26] G. Strang, *Introduction to linear algebra*, Wellesley-Cambridge Press, 2016.

[27] U. Von Luxburg, *A tutorial on spectral clustering*, Statistics and Computing (2007).