Spectral Graph Sparsification

Vedant Kumar – vsk@berkeley.edu – CS270 Survey

May 15, 2015

Abstract

Graph sparsification algorithms find sparse approximations of graphs. In this paper we survey major results in spectral sparsification, a special case in which important structural and algebraic properties of graphs are preserved. We explain a key result: that every graph has a spectral sparsifier with $O(|V| \log |V|)$ edges w.h.p, and that such sparsifiers can be found in $\tilde{O}(|E|)$ time. We finish by discussing an application of spectral sparsification to the problem of computing electrical flows in a circuit in nearly-linear time.

1 Introduction

Sparse graphs have roughly the same number of edges and vertices, up to a polylog factor ($|E| \in \tilde{O}(|V|)$). In contrast, dense graphs (such as the complete graph) permit a quadratic number of edges ($|E| \in O(|V|^2)$). If the number of vertices is fixed, many graph algorithms run significantly faster and with lower storage requirements on sparse inputs. Moreover, each edge in a sparse graph tends to contribute 'more' to the overall shape and structure of the graph. Working with sparse graphs can save time, save space, and provide insight into the nature of a graph: this makes graph sparsification interesting.

In order to evaluate sparsification algorithms, we need a way to measure 'similarity' between a graph and its sparse approximation. Section 2 discusses two different measures of graph similarity – *cut similarity* and *spectral similarity* – both of which led to the discovery of important sparsification algorithms. Section 3 contains a detailed exposition of a fast spectral sparsification algorithm based on an edge-sampling scheme. Section 4 discusses a circuit-related application.

2 Graph Similarity

In this section we explain how *cut similarity* naturally leads to *spectral similarity*, a strictly stronger measure of sparsifier quality. We also mention influential results in sparsification.

2.1 Conventions

The only graphs we consider are undirected, weighted, and connected. A graph can be described by a tuple $G = (V, E, w)$, where $w : V^2 \to \mathbb{R}$ maps edges to weights. All edges must have positive weights, so $w(u, v) = 0 \Leftrightarrow (u, v) \notin E$, allowing us to write $G = (V, w)$ w.l.o.g. For notational simplicity we define $n = |V|$ and $m = |E|$. When comparing two graphs, we always assume that they share the same vertex set. Finally, we use the notation A_k to refer to the k-th column vector of some matrix A.

2.2 Cut similarity

In this section we describe cut similarity and cite an early result in the field. Two graphs are *cut similar* if all of their cuts have approximately similar weights. To make this concrete, consider two graphs $G = (V, w)$ and $\tilde{G} = (V, \tilde{w})$. A *cut* is a subset of vertices $S \subset V$. The weight of a cut is given by the sum of the weights of edges on the cut:

$$\text{cut}_G(S) = \sum_{u \in S, v \in V - S} w(u, v)$$

G and $\tilde{G}$ are ϵ-cut similar if for all possible cuts, we have:

$$(1 - \epsilon)\text{cut}_{\tilde{G}}(S) \leq \text{cut}_G(S) \leq (1 + \epsilon)\text{cut}_{\tilde{G}}(S) \quad \forall S \subset V$$

An important early result states that every graph has a good cut similar approximation with $\tilde{O}(n)$ edges, and that this approximation can be found quickly:

THEOREM 1 (BENCZÚR-KARGER): *Let $\epsilon > 0$. $G = (V, E)$ has an ϵ-cut similar subgraph $\tilde{G}$ s.t $\tilde{m} \in O(\epsilon^{-2} n \log n)$. $\tilde{G}$ can be found in $O(m \log^3 n + \epsilon^{-2} m \log n)$ time [1].*

This version of Theorem 1 is taken from a later survey [2]. The original paper claims that an even better $O(m \log^2 n)$ time bound is possible using the union-find data structure. Regardless, this result sets the stage for future work by showing that high-quality cut-similar sparsifiers exist for *every* graph.

2.3 Spectral similarity

In this section we motivate and define spectral similarity.

The *Laplacian quadratic form* of a graph $G = (V, E, w)$ is given by the map $Q_G : \mathbb{R}^n \to \mathbb{R}$:

$$Q_G(x) = \sum_{(u,v) \in E} w(u, v)(x(u) - x(v))^2$$

Consider a cut $S \subset V$: the i-th component of its *characteristic vector* $\mathbf{1}_S$ is 1 if $v_i \in S$, and is 0 otherwise. The expression $(\mathbf{1}_S(u) - \mathbf{1}_S(v))^2$ is 1 iff (u, v) is an edge on the cut boundary of S, and is 0 otherwise. We conclude that $Q_G(\mathbf{1}_S)$ sums the weights of the edges on the cut S:

$$\text{cut}_G(S) = Q_G(\mathbf{1}_S)$$

We get cut similarity by restricting Q_G to $\{0, 1\}^n$. The graphs $(G, \tilde{G})$ are *ϵ-spectrally similar* if:

$$(1 - \epsilon)Q_{\tilde{G}}(x) \leq Q_G(x) \leq (1 + \epsilon)Q_{\tilde{G}}(x) \quad \forall x \in \mathbb{R}^n$$

There is an intuitive way to see that cut similarity is not as strong as spectral similarity. Consider a k-cycle and a k-path. We can make the k-path a k-cycle by adding an edge e'. Most cuts on $\{1..k\}$ have identical weights on both graphs because they don't touch e', so the graphs are fairly cut-similar. However, we know that the k-cycle is 2-connected while the k-path isn't. We conclude that the graphs have dissimilar λ_2's, the bipartitedness eigenvalue, implying spectral dissimilarity [3]. Apart from connectivity, spectrally similar graphs are more likely to share several other structural characteristics captured by the Laplacian quadratic form.

Spectrally similar graphs also share algebraic properties. Consider the problem of finding vertex labellings $(l, \tilde{l})$ for ϵ-spectrally similar graphs $(G, \tilde{G})$ with some of the labels fixed a priori. We can pose this as a regression problem. First solve for the unknowns in $l \in \mathbb{R}^n$ by minimizing $Q_G(l)$, then use the result to approximate a full labelling $\tilde{l}$. Intuitively, we expect to find a good approximation because $Q_G(x)$ and $Q_{\tilde{G}}(x)$ match very well over $\mathbb{R}^n$ [2].

2.4 The Laplacian quadratic form

In this section we explain some important properties of the Laplacian quadratic form. We rely on these properties heavily throughout the rest of this survey. The central object we are interested in is the $n \times n$ graph *Laplacian matrix*:

$$\begin{aligned} L_G &= D - A \\ &= \begin{cases} -w(u,v) & \text{if } u \neq v \\ \sum_z w(u,z) & \text{if } u = v \end{cases} \end{aligned}$$

Here, D is a diagonal matrix of vertex degrees and A is the adjacency matrix. We can use L_G to simplify the Laplacian quadratic form:

$$Q_G(x) = x^T L_G x$$

To see why this is true, recall that the quadratic form of a matrix A is $x^T A x = \sum_{i,j} A_{i,j} x_i x_j$. Every edge (u, v) in G contributes $w(u,v)[x_u^2 - 2x_u x_v + x_v^2]$ to the sum $Q_G(x)$. If we add up the following contributions, it becomes clear that our two definitions are equivalent:

$$\begin{aligned} i = u, j = u: &\quad w(u,v)x_u^2 \\ i = v, j = v: &\quad w(u,v)x_v^2 \\ i = u, j = v: &\quad -w(u,v)x_u x_v \\ i = v, j = u: &\quad -w(u,v)x_u x_v \end{aligned}$$

We can order symmetric PSD matrices by looking at their quadratic forms:

$$A \preceq B \Leftrightarrow x^T A x \leq x^T B x \quad \forall x \in \mathbb{R}^n$$

Now we can check if two graphs are ϵ-spectrally similar by comparing their Laplacians [1]:

$$(1-\epsilon)L_{\tilde{G}} \preceq L_G \preceq (1+\epsilon)L_{\tilde{G}}$$

The '$\preceq$' relation would be more useful if there were a simple way to compute it. By the Courant-Fischer theorem [4], the i-th eigenvalue of a matrix is given by:

$$\lambda_i(A) = \max_{S:\dim(S)=i} \min_{x \in S} \frac{x^T A x}{x^T x}$$

Note that the quadratic form of A appears in $\lambda_i(A)$. We conclude that for ϵ-spectrally similar graphs, the eigenvalues of the graph Laplacians are similar:

$$(1-\epsilon)\lambda_i(L_{\tilde{G}}) \leq \lambda_i(L_G) \leq (1+\epsilon)\lambda_i(L_{\tilde{G}}) \quad \forall i$$

This fact can be used to show that the total edge weight of a graph is roughly preserved by an ϵ-spectral approximation:

$$\mathrm{Tr}(L_G) = 2\sum_{e \in E} w(e) = \sum_i \lambda_i(L_G) \approx \mathrm{Tr}(L_{\tilde{G}}) = 2\sum_{e \in \tilde{E}} \tilde{w}(e)$$

Edges in spectral sparsifiers literally 'carry more weight' and play a greater role in maintaining the structure of the graph.

[1] We have used the fact that the Laplacian is positive semi-definite here, which is shown in Section 3.

2.5 Building spectral sparsifiers

In this section, we discuss a result which shows that every graph has a sparse, spectrally-similar cousin. We are interested in building spectral sparsifiers using edge-sampling s.t $\tilde{G} \subset G$. In order to achieve spectral similarity, we would like $L_{\tilde{G}}$ to be close to L_G in expectation. We can enforce $E[L_{\tilde{G}}] = L_G$ with the following process:

- Assign each edge a selection probability p_e (most of our effort is expended here).
- Initialize all edge weights $\tilde{w}_e$ in $\tilde{G}$ to 0: when an edge e is added to $\tilde{G}$, add w_e/p_e to $\tilde{w}_e$.
- Sample edges from G until $\tilde{G}$ is large enough.

If the edge selection probabilities are too high, $\tilde{G}$ becomes too dense. If they are too low, the spectrum of $L_{\tilde{G}}$ may not be close enough to the spectrum of L_G. The probabilities p_e should reflect the fact that some edges are more important than others.

Informally, the *conductance* of a subset $S \subset V$ is the number of edges on the cut boundary of S divided by the number of edges 'contained' by the cut. An early edge-sampling scheme worked by splitting up the vertex set of G into high-conductance components, and by prioritizing the edges between these components. This led to the following result:

THEOREM 2 (SPIELMAN-TENG): *Let $\epsilon > 0$. W.h.p $G = (V, E)$ has an ϵ-spectrally similar subgraph $\tilde{G}$ s.t $\tilde{m} \in O(\epsilon^{-2} n \log^2 n)$. $\tilde{G}$ can be found in $O(m \log^{O(1)} m)$ time [5] [2].*

Theorem 2 implies the somewhat surprising fact that arbitrary graphs have high-quality spectral sparsifiers. From here, our focus shifts to finding improved time and space bounds for the sparsification algorithm.

3 Fast Spectral Sparsification

The goal of this section is to examine the following fast spectral sparsification algorithm in detail:

THEOREM 3 (SPIELMAN-SRIVASTAVA): *Let $\epsilon > 0$. W.h.p $G = (V, E)$ has an ϵ-spectrally similar subgraph $\tilde{G}$ s.t $\tilde{m} \in O(\epsilon^{-2} n \log n)$. $\tilde{G}$ can be found in $\tilde{O}(m)$ time [6] [2].*

The Spielman-Srivastava algorithm has two stages: (1) compute edge selection probabilities, and (2) sample edges. The first step requires solving $\log m$ Laplacian systems, for which the best known time bound is $O(m \log m \log n \log \log n)$ [7] [2]. The second step can be completed in $O(\epsilon^{-2} n \log n)$ time, for a total runtime of $\tilde{O}(m)$.

3.1 Preliminaries

We need some basic facts and definitions to explain Theorem 3. Let each edge in G have some fixed (possibly arbitrary) orientation. The $(m \times n)$ *signed edge-vertex incidence matrix* is:

$$B(e, v) = \begin{cases} 1 & \text{if } e = (v, u) \\ -1 & \text{if } e = (u, v) \\ 0 & \text{otherwise} \end{cases}$$

The diagonal $(m \times m)$ *edge weight matrix* is: $W(e,e) = w(e)$. The Laplacian matrix L can be written in terms of edge weights and edge-vertex incidences:

$$L = B^T W B$$

To see why this is true, consider the 3-cycle (a triangle) where all edge weights are 1. We have:

$$B^T = \begin{pmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \quad B = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{pmatrix} \quad L = B^T B = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}$$

The i-th diagonal element of $B^T B$ is given by $B_i^T B_i$, which counts the number of edges in the neighborhood of vertex i (i.e $\sum_z w(i,z)$). For other entries $B_j^T B_i$ $(i \neq j)$, all non-zero terms in the inner product must be negative because they correspond to one of (i,j) or (j,i) (i.e $-w(i,j)$). The result is exactly the graph Laplacian.

We now verify that L is positive semi-definite (PSD):

$$x^T L x = x^T B^T W B x = ||W^{1/2} B x||_2^2 \geq 0 \quad \forall x \in \mathbb{R}^n$$

The null space of L is the subspace spanned by $\mathbf{1}$. I.e, $\ker(L) = \ker(W^{1/2}B) = \text{span}(\mathbf{1})$:

$$\begin{aligned} x^T L x = 0 &\Leftrightarrow ||W^{1/2} B x||_2^2 = 0 \\ &\Leftrightarrow x(u) - x(v) = 0 \quad \text{for all edges } (u,v) \in E \text{ since } Q_G(x) = 0 \\ &\Leftrightarrow x \text{ is constant} \end{aligned}$$

L can be diagonalized using the singular value decomposition. Let $\lambda_{1..n-1}$ be the non-zero eigenvalues and $u_{1..n-1}$ the eigenvectors:

$$L = \sum_{i=1}^{n-1} \lambda_i u_i u_i^T = U \Sigma V^*$$

The pseudo-inverse is given by:

$$L^+ = \sum_{i=1}^{n-1} \frac{1}{\lambda_i} u_i u_i^T = V \Sigma^+ U^*$$

When we are restricted to $\text{im}(L) = \text{span}(\mathbf{1})^\perp$, LL^+ is the identity:

$$\begin{aligned} LL^+ &= U \Sigma V^* V \Sigma^+ U^* = U \Sigma I \Sigma^+ U^* = U I U^* = I \quad \text{if } x \notin \ker(L) \\ L^+ L &= V \Sigma^+ U^* U \Sigma V^* = V \Sigma^+ I \Sigma V^* = V I V^* = I \quad \text{if } x \notin \ker(L) \end{aligned}$$

We need the pseudo-inverse because L is not full-rank, and $\mathbf{0}$ is not invertible.

3.2 The Projection matrix

In this section we define the Projection matrix Π and prove some facts about it. Later, we show that Π can be used to bound the spectral approximation error. To define Π, we need the $(m \times m)$ symmetric *resistance matrix* $R = BL^+B^T$, with diagonal entries $(R(e,e) = R_e)$ [2]. The $(m \times m)$ *projection matrix* is $\Pi = W^{1/2} R W^{1/2}$. We prove some properties of the projection matrix:

[2] We explain the significance of R in Section 4.

1. Π is a projection matrix.

   Proof: The fact that $\Pi^2 = \Pi$ can be checked mechanically. In the last step, we have $\Pi^2 = W^{1/2}BL^+LL^+B^TW^{1/2} = W^{1/2}BL^+B^TW^{1/2} = \Pi$, and we need the fact that LL^+ is the identity on $\text{im}(L^+)$.

2. The image of Π is equal to the image of $W^{1/2}B$.

   Proof: First note that $\text{im}(\Pi) \subseteq \text{im}(W^{1/2}B)$.

   $$\text{im}(\Pi) = \text{im}(W^{1/2}BL^+B^TW^{1/2}) \subseteq \text{im}(W^{1/2}B)$$

   Any vector $y \in \text{im}(W^{1/2}B)$ must satisfy $W^{1/2}Bx$ for some $x \notin \ker(L)$, which means:

   $$\begin{aligned}\Pi y &= W^{1/2}BL^+B^TW^{1/2}W^{1/2}Bx \\ &= W^{1/2}BL^+Lx \\ &= W^{1/2}Bx \quad \text{since } x \notin \ker(L) \\ &= y \in \text{im}(\Pi)\end{aligned}$$

   So $\text{im}(\Pi) = \text{im}(W^{1/2}B)$.

3. The eigenvalues of Π are 1 ($n-1$ copies) and 0 ($m-n+1$ copies).

   Proof: By rank-nullity on L, $\dim(\ker(L)) + \dim(\text{im}(L)) = 1 + (n-1) = n$. From this we have $\dim(\text{im}(\Pi)) = n-1$. Because Π is a projection matrix all of its eigenvalues are either 1 or 0: since the image of Π has dimension $n-1$ it must have $n-1$ non-zero ('1') eigenvalues. The remaining $m-n+1$ eigenvalues must be zeros.

4. The diagonal elements of Π are: $\Pi(e,e) = \sqrt{W(e,e)}R_e\sqrt{W(e,e)} = w(e)R_e$. We know Π is symmetric by symmetry of W and R, so we get $\Pi(e,e) = \Pi^2(e,e) = \Pi_e^T\Pi_e = ||\Pi_e||_2^2$.

3.3 Linking the Laplacian and Projection matrices

Our goal is to find a sparsifier $\tilde{G}$ s.t the quadratic forms of L and $\tilde{L}$ are close. In this section, we show that sparsifiers which keep the quadratic forms of Π and $\tilde{\Pi}$ close achieve this. The first step towards showing this is finding a symbolic expression for $\tilde{L}$. If we sample q edges to build $\tilde{G}$, we have a random $(m \times m)$ *edge sampling matrix* given by:

$$S(e,e) = \frac{\tilde{w_e}}{w_e} = (\text{times e sampled})\frac{w_e}{qp_e}\frac{1}{w_e} = \frac{(\text{times e sampled})}{qp_e}$$

Now write $\tilde{L}$ in terms of S, which controls 'how much' of each edge is preserved:

$$\tilde{L} = B^T\tilde{W}B = B^TWSB$$

In expectation, the sparsifier exactly preserves all edge weights (implying $E[\tilde{L}] = L$ as desired):

$$E\left[\frac{\tilde{w_e}}{w_e}\right] = \frac{1}{qp_e}E[(\text{times e sampled})] = \frac{1}{qp_e}qp_e = 1$$

Now we prove the main theorem in this section:

THEOREM 4 (SPIELMAN-SRIVASTAVA): If $||\Pi S\Pi - \Pi\Pi||_2 \leq \epsilon$, we have $(1-\epsilon)L \preceq \tilde{L} \preceq (1+\epsilon)L$.

By definition of the 2-norm for symmetric matrices:

$$\begin{aligned}||\Pi S\Pi - \Pi\Pi||_2 &= \sup_{y\in\mathbb{R}^m-\{\mathbf{0}\}} \frac{|y^T(\Pi S\Pi - \Pi\Pi)y|}{y^Ty} \\ &= \sup_{y\in\mathbb{R}^m-\{\mathbf{0}\}} \frac{|y^T(\Pi(S-I)\Pi)y|}{y^Ty} \\ &= \sup_{y\in\mathrm{im}(\Pi)} \frac{|y^T(\Pi(S-I)\Pi)y|}{y^Ty}\end{aligned}$$

In the last step, we only optimize over $\mathrm{im}(\Pi) \subset \mathrm{dom}(\Pi)$ since the norm must be non-negative. Now, since $y \in \mathrm{im}(\Pi)$ already, we have $\Pi y = y$ so we can drop the extra Π terms:

$$||\Pi S\Pi - \Pi\Pi||_2 = \sup_{y\in\mathrm{im}(\Pi)} \frac{|y^T(S-I)y|}{y^Ty}$$

Since we also have $y \in \mathrm{im}(W^{1/2}B)$, we know $y = W^{1/2}Bx$ for some x:

$$\begin{aligned}||\Pi S\Pi - \Pi\Pi||_2 &= \sup_{x\in\mathbb{R}^n, W^{1/2}Bx\neq\mathbf{0}} \frac{|x^TB^TW^{1/2}SW^{1/2}Bx - x^TB^TWBx|}{x^TB^TWBx} \\ &= \sup_{x\in\mathbb{R}^n, W^{1/2}Bx\neq\mathbf{0}} \frac{x^T\tilde{L}x - x^TLx}{x^TLx} \\ &\leq \epsilon \quad \text{(by assumption)}\end{aligned}$$

With some rearranging, we get:

$$\begin{aligned}\frac{x^TLx - x^T\tilde{L}x}{x^TLx} &\leq \epsilon \\ x^TLx - x^T\tilde{L}x &\leq \epsilon x^TLx \\ (1-\epsilon)x^TLx &\leq x^T\tilde{L}x\end{aligned}$$

Analogous algebra with $\frac{x^T\tilde{L}x - x^TLx}{x^TLx}$ lets us conclude $(1-\epsilon)L \preceq \tilde{L} \preceq (1+\epsilon)L$. We are now equipped with a powerful link between L and Π which simplifies the remainder of the analysis and leads to a practical edge-sampling strategy.

3.4 Bounding the expected approximation error

In this section we take a closer look at edge-selection probabilities and show how to bound the projection matrix approximation error $||\Pi S\Pi - \Pi\Pi||_2^2$ w.h.p. We use the following bound:

THEOREM 5 (RUDELSON-VERSHYNIN): Let $\hat{p}$ be a probability distribution over $\Omega \subset \mathbb{R}^d$ s.t $\sup_{y\in\Omega} ||y||_2 \leq M$ and $||E_{\hat{p}}[yy^T]||_2 \leq 1$. If $y_{1..q}$ are drawn independently from $\hat{p}$, then [2] [6] [8]:

$$E\left[\left|\frac{1}{q}\sum_{i=1}^{q} y_iy_i^T - E[yy^T]\right|_2\right] \leq \min\left(CM\sqrt{\frac{\log q}{q}}, 1\right)$$

The probabilities p_e are weight-proportional to the *effective resistances* R_e from the resistance matrix. Given that $\sum_e w_eR_e = \mathrm{Tr}(\Pi) = n-1$, we set $p_e = \frac{w_eR_e}{n-1}$ to get a normalized probability

distribution. Sampling q edges (with replacement) corresponds to sampling columns of Π, giving:

$$\Pi S \Pi = \sum_e S(e,e)\Pi_e \Pi_e^T = \sum_e \frac{(\text{times e sampled})}{qp_e}\Pi_e \Pi_e^T = \frac{1}{q}\sum_{i=1}^{q} y_i y_i^T$$

Where the y_i are i.i.d from $\hat{y} = \frac{1}{\sqrt{p_e}}\Pi_e$ s.t $\Pr[\hat{y} = y_e] = p_e$. We now have the first term we need for the bound. The expected value of yy^T over $\hat{y}$ is $\sum_e p_e \frac{1}{\sqrt{p_e}}\frac{1}{\sqrt{p_e}}\Pi_e\Pi_e^T = \Pi^2 = \Pi$, which gives us the second term we need. We verify that $||E[yy^T]||_2 = ||\Pi||_2 \leq 1$ (by definition, the spectral or 2-norm of Π is its largest singular value, 1). Finally, we verify that y has an upper bound:

$$||y||_2 = \frac{1}{p_e}\sqrt{\Pi(e,e)} = \sqrt{\frac{n-1}{w_e R_e} w_e R_e} = \sqrt{n-1} = M$$

We pick $q \in O(\frac{C^2}{\epsilon^2} n \log n)$ and apply Theorem 5 to get:

$$E[|\Pi S \Pi - \Pi\Pi|_2] = E\left[\left|\frac{1}{q}\sum_{i=1}^{q} y_i y_i^T - E[yy^T]\right|_2\right] \leq \frac{\epsilon}{2}$$

Which holds for large n and when $\epsilon \geq \frac{1}{\sqrt{n}}$. Markov's inequality tells us that the probability of the approximation error being greater than ϵ is upper-bounded by $E[|\Pi S\Pi - \Pi\Pi|_2]/\epsilon = \frac{1}{2}$. In practice we expect the approximation error to be tightly concentrated around $\frac{\epsilon}{2}$.

3.5 Computing the edge selection probabilities

In this section we discuss how to find a good set of edge selection probabilities. Recall that $p_e \approx w_e R_e$, and that the effective resistance matrix is given by $R = BL^+B^T$. It would be slow to compute R exactly. Fortunately, an α-approximation of R can be used in Theorem 5 with the only side-effect of pushing the expected approximation error to $\frac{\alpha\epsilon}{2}$.

To find a good approximation of R, we use a fact proven in Section 4: the effective resistances R_e are just pairwise distances of vectors in the space:

$$W^{1/2}BL^+\mathbf{1}_v \quad \forall v \in V$$

These distances are more-or-less preserved if we project this space onto a subspace spanned by $O(\log m)$ random vectors:

THEOREM 6 (JOHNSON-LINDENSTRAUSS): Fix vectors $v_{1..m} \in \mathbb{R}^n$. Let $\epsilon > 0$, $k \geq 24\epsilon^{-2}\log m$, and $Q_{k\times m}$ be a random matrix with entries $\pm k^{-1/2}$. Then with probability $\geq 1 - \frac{1}{m}$ [6] [9]:

$$(1-\epsilon)||v_i - v_j||_2^2 \leq ||Qv_i - Qv_j||_2^2 \leq (1+\epsilon)||v_i - v_j||_2^2 \quad \forall (i,j)$$

Let $\tilde{Z} \approx QW^{1/2}BL^+$. We can use a fast solver for Laplacian systems to compute its rows. Spielman-Srivastava use the solver from an earlier paper [5]. A later survey recommends the solver by Koutis et. al [2] [7].

3.6 Overview

In this section we summarize the Spielman-Srivastava algorithm. We also mention one last spectral sparsification algorithm. It produces a sparsifier with a linear number of edges in cubic time. Standard Spielman-Srivastava can be run as follows:

- Compute an approximate effective resistance matrix $\tilde{Z} \approx QW^{1/2}BL^{+}$ by using a fast solver for Laplacian systems.

  The entries of the matrix Q are $\pm k^{-1/2}$, where the sign is positive with probability $1/2$. We assume efficient representations for W and B are used, since they contain only m and $2m$ entries respectively. It is not necessary to compute SVD(L) to find L^{+}: we assume the Laplacian solver computes the pseudo-inverse efficiently. Each one of the $O(\log m)$ rows of $\tilde{Z}$ can be computed by running $\tilde{z}_i = \text{LSolve}(L, [QW^{1/2}B]_i, \delta)$ for some error parameter δ.

- Compute $p_e = \frac{w_e||\tilde{Z}(\mathbf{1}_u - \mathbf{1}_v)||_2^2}{n-1}$ for each edge. Sample q edges based on the distribution p_e.

  Note that edges may be sampled efficiently using the array $P = [p_{e_1}, p_{e_1} + p_{e_2}, ..., \sum_e p_e]$. Once P is constructed, we can sample an edge by generating a random number $r \in [0, 1]$ and binary searching for r in P.

There is another sparsification algorithm by Batson, Spielman, and Srivastava which produces $O(n)$-edge spectral sparsifiers in $O(mn^3)$ time [10]. We mention it here for two reasons. First, it exhibits a classic space-time tradeoff (i.e it has an asymptotically higher runtime but produces sparser results). Second, it implies that every graph has a high-quality spectral sparsifier with a linear number of edges, which is very surprising.

4 Computing Electrical Flows

In this section we explain why R is called the effective resistance matrix, and show how to use $\tilde{Z}$ to approximate the resistance between any pair of vertices in a circuit. In conjunction with Ohm's law, this information gives us a way to compute voltage drops between nodes in a circuit as well as current flow.

Consider a graph G where the edges represent wires and the vertices represent nodes in a circuit. The edge weights correspond to conductances (i.e inverse resistances). Following the convention in [6], let $i_{ext}(u)$ denote the amount of externally-supplied current at each node, let $i(e)$ denote the current flow along each edge, and let $v(u)$ denote the voltage at each node. The current flowing into a node equals the current flowing out:

$$B^T i = i_{ext}$$

By Ohm's law, $i = v/r = v\sigma$, so:

$$i = WBv$$

This gives $i_{ext} = B^T(WBv) = Lv$. As long as we don't have a shorted circuit, i_{ext} is not in $\ker(L)$ and we may write $v = L^{+}i_{ext}$. To compute effective resistances between any two nodes (u, v), we inject a unit current at u and extract the current at v:

$$v_v - v_u = (\mathbf{1}_v - \mathbf{1}_u)^T v$$

Because $i_{ext} = (\mathbf{1}_v - \mathbf{1}_u)$, this expression gives us a row of $B^T L^{+} B = R$, the matrix of effective resistances between all pairs of nodes in the circuit. By the work done in the previous section, we know that it is possible to compute $\tilde{Z}$ in $\tilde{O}(m)$ time. We can approximate effective resistances between any pair of vertices by taking $||\tilde{Z}(\mathbf{1}_u - \mathbf{1}_v)||_2^2$, which is fast ($O(\log m)$ time).

References

[1] András A Benczúr and David R Karger. Approximating st minimum cuts in $\tilde{O}(n^2)$ time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 47–55. ACM, 1996.

[2] Joshua Batson, Daniel A Spielman, Nikhil Srivastava, and Shang-Hua Teng. Spectral sparsification of graphs: theory and algorithms. *Communications of the acm*, 56(8):87–94, 2013.

[3] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[4] Yasuhiko Ikebe, Toshiyuki Inagaki, and Sadaaki Miyamoto. The monotonicity theorem, cauchy's interlace theorem, and the courant-fischer theorem. *American Mathematical Monthly*, pages 352–354, 1987.

[5] Daniel A Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.

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

[7] Ioannis Koutis, Alex Levin, and Richard Peng. Improved spectral sparsification and numerical algorithms for sdd matrices. In *STACS'12 (29th Symposium on Theoretical Aspects of Computer Science)*, volume 14, pages 266–277. LIPIcs, 2012.

[8] Mark Rudelson and Roman Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *Journal of the ACM (JACM)*, 54(4):21, 2007.

[9] Dimitris Achlioptas and Frank McSherry. Fast computation of low rank matrix approximations. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 611–618. ACM, 2001.

[10] Joshua Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012.