DOI:10.1145/2492007.2492029

Spectral Sparsification of Graphs: Theory and Algorithms

By Joshua Batson, Daniel A. Spielman, Nikhil Srivastava, and Shang-Hua Teng

Abstract

Graph sparsification is the approximation of an arbitrary graph by a sparse graph.

We explain what it means for one graph to be a spectral approximation of another and review the development of algorithms for spectral sparsification. In addition to being an interesting concept, spectral sparsification has been an important tool in the design of nearly linear-time algorithms for solving systems of linear equations in symmetric, diagonally dominant matrices. The fast solution of these linear systems has already led to breakthrough results in combinatorial optimization, including a faster algorithm for finding approximate maximum flows and minimum cuts in an undirected network.

1. INTRODUCTION

A sparse graph is one whose number of edges is reasonably viewed as being proportional to its number of vertices. In contrast, the complete graph on n vertices, with about $n^2/2$ edges, is the paradigmatic dense graph. Sparse graphs are often easier to handle than dense ones. Most graph algorithms run faster, sometimes by orders of magnitude, when there are fewer edges, and the graph itself can be stored more compactly. By approximating a dense graph of interest by a suitable sparse one, one can save time and space.

We will work with weighted graphs, where the weights might represent capacities, conductance, similarity, or just coefficients in a linear system. In a sparse graph, all of the edges can be important for a graph's structure. In a tree, for example, each edge provides the only path between its endpoints. Not so in a dense graph, where some edges will serve similar functions. A collection of low-weight edges connecting two clusters of vertices in a graph might be approximable by a single high-weight edge connecting vertices representative of those clusters. Sparsification can be viewed as a procedure for finding a set of representative edges and weighting them appropriately.

What exactly do we mean by sparse? We would certainly consider a graph sparse if its average degree were less than 10, and we would probably consider a graph sparse if it had one billion vertices and average degree one hundred. We formalize the notion of sparsity in the usual analysis-of-algorithms way by considering infinite families of graphs, and proclaiming sparse those whose average degrees are bounded by some constant, or perhaps by a polynomial in the logarithm of their number of vertices.

One may at first think that sparsification is unnecessary, as common wisdom holds that all large real-world graphs are sparse. While this may be true of natural graphs such as social networks, it is not true of the graphs that arise inside algorithms. Many algorithms involve the construction of graphs that are dense, even when solving problems on graphs that are sparse. Moreover, the common wisdom may be an artifact of the difficulty of storing and manipulating a large dense graph. Improvements in sparsification may one day ameliorate these difficulties.

The use of sparse graphs to approximate dense ones is not unique to algorithm design. In a parallel computer, for instance, information needs to be able to flow from any processor to any other. Hardwiring all those pairwise connections would be physically difficult, so a sparse graph simulating the connectivity properties of the complete graph is needed. The hypercube graph plays this role in the CM5, built by Thinking Machines.[25] Intuitively, the hypercube has no "bottlenecks." Formally, the (weighted) hypercube is a good spectral sparsifier for the complete graph defined on its nodes. We have shown that *every* graph has a very sparse spectral approximation, with constant average degree.

2. NOTIONS OF SIMILARITY

A few conventions: we specify a weighted graph by a 3-tuple, $G = (V, E, w)$, with vertex set $V = \{1, \ldots, n\}$, edge set $E \subseteq \{(u, v) \mid u, v \in V\}$, and weights $w_{(u,v)} > 0$ for each $(u, v) \in E$. All graphs will be undirected and weighted, unless otherwise stated. We sometimes express a graph simply by $G = (V, w)$, as E can be defined implicitly by setting $w_{(u,v)} = 0$ for all $(u, v) \notin E$. We will always write n for the number of vertices in a graph and m for the number of edges. When measuring the similarity between two graphs, we will always assume that they have the same set of vertices.

2.1. Cut similarity

The notion of cut similarity of graphs was first considered by Benczúr and Karger[8] as part of an effort to develop fast algorithms for the minimum cut and maximum flow problems. In these problems, one is interested in the sum of the weights of edges that are cut when one divides the vertices of a graph into two pieces. Two weighted graphs on the same vertex set are cut-similar if the sum of the weights of the edges cut is approximately the same in each such division.

To write this symbolically, we first observe that a division of the vertices into two parts can be specified by identifying

A previous version of the paper, "Twice-Ramanujan Sparsifiers," was published in the *Proceedings of the 41st Annual ACM Symposium on the Theory of Computing* (2009), 255–262.

the subset of vertices in one part. For a weighted graph $G = (V, w)$ and a subset of vertices $S \subset V$, we define

$$cut_G(S) \stackrel{\text{def}}{=} \sum_{u \in S,\ v \in V-S} w_{(u,v)}.$$

We say that $G = (V, w)$ and $\tilde{G} = (V, \tilde{w})$ are *σ-cut similar* if

$$cut_{\tilde{G}}(S) / \sigma \le cut_G(S) \le \sigma \cdot cut_{\tilde{G}}(S),$$

for all $S \subset V$. Surprisingly, every graph is cut-similar to a graph with average degree $O(\log n)$, and that graph can be computed in polylogarithmic time.

THEOREM 1 (BENCZÚR-KARGER). *For all $\epsilon > 0$, every $G = (V, E, w)$ has a $(1 + \epsilon)$-cut similar graph $\tilde{G} = (V, \tilde{E}, \tilde{w})$ such that $\tilde{E} \subseteq E$ and $|\tilde{E}| = O(n \log n/\epsilon^2)$. Moreover $\tilde{G}$ can be computed in $O(m \log^3 n + m \log n/\epsilon^2)$ time.*

The sizes of cuts in a graph tell us a lot about its structure—for starters, the weighted degrees of vertices are given by cuts of size $|S| = 1$. Most ways of defining a cluster of vertices in a graph involve comparing the number of edges in the cut defined by the set of vertices to the number of edges internal to that set.

2.2. Spectral similarity

Motivated by problems in numerical linear algebra and spectral graph theory, Spielman and Teng[34] introduced a notion of *spectral similarity* for two graphs. We will first describe it as a generalization of cut similarity.

Given a weighted graph $G = (V, w)$, we define the Laplacian quadratic form of G to be the function Q_G from $\mathbb{R}^V$ to $\mathbb{R}$ given by

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

If S is a set of vertices and x is the characteristic vector of S (1 inside S and 0 outside), then it is easy to see that

$$Q_G(x) = cut_G(S).$$

We say two graphs $G = (V, w)$ and $\tilde{G} = (V, \tilde{w})$ are *σ-spectrally similar* if

$$Q_{\tilde{G}}(x) / \sigma \le Q_G(x) \le \sigma \cdot Q_{\tilde{G}}(x), \quad \text{for all } x \in \mathbb{R}^V. \tag{1}$$

Thus, cut similarity can be viewed as the special case of spectral similarity in which we only consider vectors x that take values in $\{0, 1\}$.

It is possible to construct graphs that have very similar cuts, but which are highly dissimilar from the spectral perspective; for instance, the n-vertex path is 2-cut similar but only n-spectrally similar to the n-vertex cycle. Although spectral similarity is strictly stronger than cut similarity, it is easier to check if two graphs are spectrally similar. In particular, one can estimate the spectral similarity of two graphs to precision ϵ in time polynomial in n and $\log(1/\epsilon)$, but it is NP-hard to approximately compute the cut-similarity of two graphs.

Graphs that are spectrally similar share many algebraic properties. For example, the effective resistance distances between all pairs of vertices are similar in spectrally similar graphs. The effective resistance distance is defined by viewing each edge in a graph as a resistor: an edge of weight w becomes a resistor of resistance $1/w$. The entire graph is then viewed as a resistive circuit, and the effective resistance between two vertices is just the electrical resistance in the network between them. It equals the potential difference induced between the vertices when a unit current is injected at one and extracted at the other. In terms of the Laplacian quadratic form, the effective resistance between vertices u and v may be written as

$$R_{(u,v)} = \left(\min_{x: x(u) = 1, x(v) = 0} Q_G(x) \right)^{-1},$$

an identity which follows from the well-known energy minimizing property of electrical flows.

The Laplacian quadratic form provides a natural way to solve regression problems on graphs. In these problems, one is told the values of x on some subset of the nodes S and is asked to infer the values on the remaining nodes. One approach to solving these problems is to view the known values as voltages that have been fixed, and the values at the other nodes as the induced voltages. That is, one seeks the vector x that minimizes $Q_G(x)$ while agreeing with the known values.[36] One can show that if two graphs are spectrally similar, then the solutions to all such regression problems on the graphs will be similar as well.

The problems of regression and computing effective resistances are special cases of the problem that motivated the definition of spectral similarity: the solution of linear equations in Laplacian matrices. The Laplacian quadratic form can be written as

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

where L_G is the Laplacian matrix of G. The Laplacian matrix of a graph $G = (V, w)$ is defined by

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

The problem of solving systems of linear equations in Laplacian matrices arises in many areas of computational science and optimization. In fact, the spectral similarity measure is *identical* to the concept of relative condition number in numerical linear algebra. If two graphs are spectrally similar, then through the technique of preconditioning one can use solutions to linear equations in the Laplacian matrix of one graph to solve systems of linear equations in the Laplacian of the other.

2.3. Spectral similarity of matrices

For two symmetric matrices A and B in $\mathbb{R}^{n \times n}$, we write $A \preccurlyeq B$ to indicate that

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

We say A and B are *σ-spectrally similar* if

$$B/\sigma \preccurlyeq A \preccurlyeq \sigma \cdot B. \quad (2)$$

We have named this relation spectral similarity because it implies that the two matrices have similar eigenvalues. The Courant-Fisher Theorem tells us that

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

Thus, if $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of A and $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n$ are the eigenvalues of B, then for all i, $\lambda_i/\sigma \leq \tilde{\lambda}_i \leq \sigma \cdot \lambda_i$.

Using this notation, we can now write inequality (1) as

$$L_{\tilde{G}}/\sigma \preccurlyeq L_G \preccurlyeq \sigma \cdot L_{\tilde{G}}. \quad (3)$$

That is, two graphs are σ-spectrally similar if their Laplacian matrices are. We remark that the Laplacian matrix of a graph is (i) *symmetric*, that is, $L_G = L_G^T$, (ii) *positive semi-definite*, that is, all eigenvalues of L_G are non-negative, and (iii) *(weakly) diagonally dominant*, that is, for all i, $L_G(i,i) \geq \sum_{j \neq i} |L_G(i,j)|$. From consideration of the Laplacian quadratic form, it is easy to verify that if G is connected, then the null space of L_G is just the span of the all 1's vector. Thus all connected graphs have the same Laplacian null space and exactly one zero eigenvalue.

2.4. Distance similarity

It is worth mentioning an interesting alternative to cut- and spectral-similarity. If one assigns a length to every edge in a graph, then these lengths induce a shortest-path distance between every pair of vertices. We say that two different graphs on the same vertex set are *σ-distance similar* if the distance between each pair of vertices in one graph is within a multiplicative factor of σ of the distance between the corresponding pair of vertices in the other graph. Formally, if G and $\tilde{G}$ are the graphs and if $\text{dist}_G(u, v)$ is the distance between vertices u and v in G, then G and $\tilde{G}$ are *σ-distance similar* if for all $u, v \in V$,

$$\text{dist}_G(u,v)/\sigma \leq \text{dist}_{\tilde{G}}(u,v) \leq \sigma \cdot \text{dist}_G(u,v).$$

When $\tilde{G}$ is a subgraph of G, the inequality $\text{dist}_G(u,v) \leq \text{dist}_{\tilde{G}}(u,v)$ is automatically satisfied. Peleg and Ullman[29] defined a *t*-spanner of a graph G to be a subgraph such that for all $u, v \in V$,

$$\text{dist}_{\tilde{G}}(u,v) \leq t \cdot \text{dist}_G(u,v).$$

They were interested in finding sparse *t*-spanners. It has been shown[4] that every weighed graph has a $(2t+1)$-spanner with $O(n^{1+1/t})$ edges. The most extreme form of a sparse spanner is the *low stretch spanning tree*, which has only $n - 1$ edges, but which only approximately preserves distances on average,[1] up to polylogarithmic distortion.

3. FINDING SPARSE SUBSTITUTES

A (σ, d)-*spectral sparsifier* of a graph G is a graph $\tilde{G}$ satisfying

1. $\tilde{G}$ is σ-spectrally similar to G
2. The edges of $\tilde{G}$ consist of reweighted edges of G
3. $\tilde{G}$ has at most $d|V|$ edges

Since spectral similarity implies that the total edge weight of a graph is preserved, the spectral sparsifier can only have fewer edges than G if those edges have larger weights.

In this section, we begin by discussing sparsifiers of the complete graph, which have been known for some time. We then describe a sequence of ever-stronger existence theorems, culminating in the statement that any graph G has a $(1+\epsilon, 4/\epsilon^2)$-spectral sparsifier for every $\epsilon \in (0, 1)$.

3.1. Cliques have constant-degree sparsifers

To warm up, let us first examine the quality of the hypercube as a spectral sparsifier of the complete graph. Assume for convenience that n is a power of two. Let G be the complete graph on n vertices. All the non-zero eigenvalues of L_G equal n, so for every unit vector x orthogonal to the all-1s vector,

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

The non-zero eigenvalues of the Laplacian of the hypercube with n vertices in which every edge has weight 1 are $(2, \ldots, 2\log n)$. Let H be this hypercube, but with edge weights $n/(2\sqrt{\log n})$. The non-zero eigenvalues of the Laplacian of H are then

$$(n/\sqrt{\log n}, \ldots, n\sqrt{\log n})$$

which implies that for every unit vector x orthogonal to the all-1s vector,

$$x^T L_H x \in \left[n/\sqrt{\log n}, n\sqrt{\log n} \right].$$

Thus, H is a $(\sqrt{\log n}, \log n/2)$-spectral sparsifier of the n-clique.

In fact, the complete graph has much better spectral sparsifiers. Consider the Ramanujan graphs,[26, 27] which are d-regular graphs all of whose non-zero Laplacian eigenvalues lie between $d - 2\sqrt{d-1}$ and $d + 2\sqrt{d-1}$. If we let $\tilde{G}$ be a Ramanujan graph with edge weight n/d, then for every unit vector x orthogonal to the all-1s vector,

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

Thus, $\tilde{G}$ is a $\left((1 - 2\sqrt{d-1}/d)^{-1},\ d/2\right)$-spectral sparsifier of the complete graph G.

3.2. Sampling and decomposition: every graph has a good sparsifier

Ramanujan graphs are members of the family of *expander graphs*. These are sparse graphs in which every subset of vertices has a significant fraction of its edges connecting to vertices outside the subset (see below for details). As with the hypercube and Ramanujan graphs, we can show that every expander can be rescaled to be a good spectral sparsifier of the complete graph.

It is well known that random sparse graphs are usually good expanders (see Bollobás[9] and Friedman[17]). Therefore, one can obtain a good spectral sparsifier of the complete graph by random sampling. Spielman and Teng[34] took this view one step further to show that graph sampling can be

used to obtain a good spectral sparsifier for every graph. Their construction was strongly motivated by the work of Benczúr and Karger[8] and Achlioptas and McSherry.[2]

The sampling procedure involves assigning a probability $p_{u,v}$ to each edge $(u, v) \in G$, and then selecting edge (u, v) to be in the graph $\tilde{G}$ with probability $p_{u,v}$. When edge (u, v) is chosen to be in the graph, we multiply its weight by $1/p_{u,v}$.

This procedure guarantees that

$$\mathbf{E}[L_{\tilde{G}}] = L_G.$$

The key step in this approach is to determine the sampling probability $p_{u,v}$ for each edge; there is a tension between choosing small $p_{u,v}$ to generate a sparser $\tilde{G}$ and choosing larger $p_{u,v}$ to more accurately approximate G. Spielman and Teng recognized that some edges are more essential than others, and used a graph decomposition process to implicitly identify these edges and set the sampling probabilities accordingly.

Conductance and graph decomposition. For an unweighted graph $G = (V, E)$ and disjoint subsets $S, T \subset V$, we let $E(S, T)$ denote the set of edges in E connecting one vertex of S with one vertex of T. We define Vol $(S) = \sum_{i \in S} d_i$ and observe that Vol $(V) = 2|E|$. We define the *conductance* of a set S of vertices to be

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

and we define

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

The *normalized Laplacian* matrix of a graph (see [14]), is defined to be

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

where D is the diagonal matrix whose u-th entry is d_u. The discrete version[3, 31] of Cheeger's[11] inequality (Theorem 2) relates the second eigenvalue of the normalized Laplacian to the conductance of a graph.

Theorem 2 (Discrete Cheeger Inequality).

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

We define a *decomposition* of G to be a partition of V into sets $(A_1, \ldots, A_k)$, for some k. The *boundary* of a decomposition $(A_1, \ldots, A_k)$ is then the set of edges between different vertex sets in the partition:

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

We say that the decomposition is a ϕ*-decomposition* if $\Phi_{G(A_i)} \geq \phi$ for all i, where $G(A_i)$ denotes the subgraph induced on A_i. It is a λ*-spectral decomposition* if the smallest non-zero normalized Laplacian eigenvalue of $G(A_i)$ is at least λ, for all i. By Cheeger's inequality, every ϕ-decomposition is a $(\phi^2/2)$-spectral decomposition.

The following two theorems (see Spielman and Teng[34]) together imply that for all ϵ, every graph has a $((1 + \epsilon), O(\epsilon^{-2} \log^7 n))$-spectral sparsifier.

Theorem 3 (Spectral Decomposition). *Every G has an $\Omega(\log^{-2} n)$-spectral decomposition with boundary size at most $|E|/2$.*

Theorem 4 (Sampling works for Expanders). *Suppose $\epsilon \in (0, 1/2)$ and $G = (V, E)$ is an unweighted graph with smallest non-zero normalized Laplacian eigenvalue at least λ. Let $\tilde{G} = (V, \tilde{E}, \tilde{w})$ be a graph obtained by sampling the edges of G with probabilities*

$$p_e = \min(1, C/\min(d_u, d_v)) \qquad \text{for each edge } e = (u,v),$$

where

$$C = \Theta((\log n)^2 (\epsilon\lambda)^{-2})$$

and setting weights $\tilde{w}_{(e)} = 1/p_e$ for $e \in \tilde{E}$. Then, with probability at least $1/2$, $\tilde{G}$ is a $(1 + \epsilon)$-spectral approximation of G, and the average degree of $\tilde{G}$ is $O((\log n)^2 (\epsilon\lambda)^{-2})$

To construct a spectral sparsifier of an arbitrary unweighted graph, we first apply Theorem 3 to find a $\Omega(\frac{1}{\log^2 n})$-spectral decomposition of the graph in which the boundary has at most half the edges. We then sparsify each of the components by random sampling, and we sparsify the graph formed by the boundary edges recursively. Adding the sparsifiers obtained yields a sparsifier for the original graph, as desired.

3.3. Sampling by effective resistance

By using effective resistances to define the edge sampling probabilities p_e, Spielman and Srivastava[32] proved that every graph has a $((1 + \epsilon), O(\log n/\epsilon^2))$-spectral sparsifier. These spectral sparsifiers have a similar number of edges to the cut sparsifiers described in Theorem 1, and many fewer edges than those produced by Spielman and Teng[34]. We define R_e, the effective resistance of an edge e, to be the effective resistance between its endpoints. It is well-known that R_e is proportional to the commute time between the end-vertices of e,[10] and is equal to the probability that e appears in a random spanning tree of G. Spielman and Srivastava proved that sampling with edge probability p_e proportional to $w_e R_e$ is the "right" distribution for creating spectral sparsifiers.

Theorem 5 (Sampling by Effective Resistance). *For any weighted graph $G = (V, E, w)$ and $0 < \epsilon \leq 1$, let $\tilde{G}$ be the graph obtained by the following random process:*

> Set $q = 8n \log n/\epsilon^2$. Choose a random edge of G with probability p_e proportional to $w_e R_e$, and add e to the edge set of $\tilde{G}$ with weight w_e/qp_e. Take q samples independently with replacement, summing weights if an edge is chosen more than once.

Then with probability at least $1/2$, $\tilde{G}$ is a $(1 + \epsilon)$-spectral approximation of G.

The proof of Theorem 5 is matrix-analytic. We begin by observing that the Laplacian matrix of G can be expressed as a sum of outer products of vectors:

$$L_G = \sum_{(u,v)\in E} w_{(u,v)}(e_u - e_v)(e_u - e_v)^T,$$

where e_u denotes the elementary unit vector in direction u. Since the edges in $\tilde{G}$ are a subset of the edges in G, its Laplacian may also be expressed as a (differently weighted) sum of the same outer products:

$$L_{\tilde{G}} = \sum_{(u,v)\in E} \tilde{w}_{(u,v)}(e_u - e_v)(e_u - e_v)^T.$$

Suppose the non-zero eigenvalue-and-eigenvector pairs of L_G are $(\lambda_1, u_1), \ldots, (\lambda_{n-1}, u_{n-1})$. Then we can write

$$L_G = \sum_{i=1}^{n-1} \lambda_i u_i u_i^T.$$

Let L_G^+ be the Moore-Penrose Pseudoinverse of L_G, that is,

$$L_G^+ = \sum_{i=1}^{n-1} u_i u_i^T / \lambda_i.$$

Then $L_G L_G^+ = L_G^+ L_G = \sum_i u_i u_i^T \stackrel{\text{def}}{=} \Pi$ is the projection matrix onto the span of $\{u_i\}$.

The key to the analysis of Theorem 5, and the improved construction of Section 3.4, is the observation that

$$L_G/\sigma \preceq L_{\tilde{G}} \preceq \sigma L_G \Leftrightarrow \prod/\sigma \preceq L_G^{+/2} L_{\tilde{G}} L_G^{+/2} \preceq \sigma \prod,$$

where $L_G^{+/2}$ is the square root of L_G^+. We will show that the sampling procedure described is likely to satisfy this latter condition. To this end, define random variables $\{s_e\}_{e\in E}$ to capture the outcome of the sampling procedure:

$$s_e \stackrel{\text{def}}{=} \frac{\tilde{w}_e}{w_e} = \frac{(\#\text{ of times } e \text{ is sampled})}{qp_e}. \quad (4)$$

Then

$$L_{\tilde{G}} = \sum_{(u,v)\in E} s_{(u,v)} w_{(u,v)} (e_u - e_v)(e_u - e_v)^T$$

and $\mathbf{E}[s_{(u,v)}] = 1$ for all $(u,v) \in E$, whence $\mathbf{E}[L_{\tilde{G}}] = L_G$. We now write:

$$L_G^{+/2} L_{\tilde{G}} L_G^{+/2} = \sum_{(u,v)\in E} s_{(u,v)} w_{(u,v)} L_G^{+/2}(e_u - e_v)(e_u - e_v)^T L_G^{+/2}$$
$$= \frac{1}{q} \sum_{i\le q} Y_i Y_i^T,$$

where the Y_i are i.i.d. random vectors sampled from the distribution

$$Y = \frac{1}{\sqrt{p_{(u,v)}}} \left(w_{(u,v)}^{1/2} L_G^{+/2}(e_u - e_v)\right) \quad \text{with probability } p_{(u,v)}.$$

Notice that $\mathbf{E}[YY^T] = \Pi$ so the expectation of each term is correct. To analyze the sum, we apply the following concentration theorem for sums of independent rank one matrices due to Rudelson.[30]

THEOREM 6 (RUDELSON). *Let p be a probability distribution over $\Omega \subset \mathbb{R}^d$ such that $\sup_{y\in\Omega} ||y||_2 \le M$ and $||\mathbf{E}[yy^T]|| \le 1$. Let $y_1, \ldots, y_q$ be independent samples drawn from p with replacement. Then,*

$$\mathbf{E}\left[\left\|\frac{1}{q}\sum_{i=1}^{q} y_i y_i^T - \mathbf{E}[yy^T]\right\|_2\right] \le \min\left(8M\sqrt{\frac{\log q}{q}}, 1\right).$$

To optimize the application of Rudelson's theorem, we choose the $\{p_{(u,v)}\}$ so that all possible values of Y have the same norm, that is,

$$\|Y\| = \left\|\frac{1}{\sqrt{p_{(u,v)}}}(w_{(u,v)}^{1/2} L_G^{+/2}(e_u - e_v))\right\| = \gamma$$

for some fixed γ, for all $(u,v) \in E$. An elementary calculation reveals that the value of γ that causes the sum of the probabilities to be one is $\sqrt{n-1}$. Thus if we sample according to probabilities

$$p_{u,v} = \frac{\left\|w_{(u,v)}^{1/2} L_G^{+/2}(e_u - e_v)\right\|^2}{n-1},$$

then we can take $M = \sqrt{n-1}$ in Theorem 6. This tells us that $q = O(n \log n/\epsilon^2)$ samples are sufficient to obtain a $(1+\epsilon)$-spectral approximation with high probability, from which Theorem 5 follows.

As stated earlier, the probabilities we have chosen for sampling edges have a natural meaning:

$$\left\|w_{(u,v)}^{1/2} L_G^{+/2}(e_u - e_v)\right\|^2 = w_{(u,v)}(e_u - e_v)L_G^+(e_u - e_v) = w_{(u,v)} R_{(u,v)},$$

where

$$R_{(u,v)} \stackrel{\text{def}}{=} (e_u - e_v)L_G^+(e_u - e_v)$$

is the effective resistance between the vertices u and v.

3.4. Twice-Ramanujan sparsifiers

In a nutshell, Spielman and Srivastava first reduced the sparsification of $G = (V, E, w)$ to the following algebraic problem: compute scalars $\{s_{u,v} \ge 0 | (u,v) \in E\}$ such that $\tilde{E} = \{e | s_{u,v} > 0\}$ has small cardinality, and

$$(1-\epsilon)\prod \preccurlyeq L_G^{+/2} L_{\tilde{G}} L_G^{+/2} \le (1+\epsilon)\prod.$$

They then applied sampling, based on effective resistances, to generate the $\{s_{u,v}\}$ and $\tilde{E}$.

Batson et al.[7] gave a deterministic polynomial-time algorithm for computing $\{s_e\}$ and $\tilde{E}$, and obtained the following theorem, which is essentially the best possible result for spectral sparsification.

THEOREM 7 (BATSON-SPIELMAN-SRIVASTAVA). *For every $d > 1$, every undirected, weighted n-node graph $G = (V, E, w)$ has a*

$$\left(\frac{\sqrt{d}+1}{\sqrt{d}-1}, d\right) \text{- spectral sparsifier.}$$

In particular, G has a $((1+2\epsilon), 4/\epsilon^2)$-spectral sparsifier, for every $0 < \epsilon < 1$.

At the heart of their construction is the following purely linear algebraic theorem, which may be shown to imply Theorem 7 by an argument similar to that in Section 3.3.

THEOREM 8. *Suppose $d > 1$ and $v_1, \ldots, v_m \in \mathbb{R}^n$ satisfy $\sum_{i=1}^m v_i v_i^T = I_n$, where I_n is the $n \times n$ identity matrix. Then, there exist scalars $s_i \ge 0$ with $|\{i : s_i \ne 0\}| \le dn$ such that*

$$I_n \preccurlyeq \sum_{i=1}^{m} s_i v_i v_i^T \preccurlyeq \left(\frac{\sqrt{d}+1}{\sqrt{d}-1}\right)^2 \cdot I_n$$

The proof for Theorem 8 builds the sum $\sum_i s_i v_i v_i^T$ iteratively, by adding one vector at a time. For $\tilde{m} = dn$, it chooses a sequence of vectors $\pi(1), \ldots, \pi(\tilde{m})$ and weights $s_{\pi(1)}, \ldots, s_{\pi(\tilde{m})}$, which in turn defines a sequence of matrices $0 = A_0, \ldots, A_{\tilde{m}}$, where $A_t = \sum_{i=1}^{t} s_{\pi(i)} v_{\pi(i)} v_{\pi(i)}^T$. Observe that $A_t = A_{t-1} + s_{\pi(t)} v_{\pi(t)} v_{\pi(t)}^T$ and we always have $A_t \succeq A_{t-1}$. The goal is to control the eigenvalues of A_t at each step and guarantee that they grow with t in a *steady* manner, so that the final matrix $A_{\tilde{m}} = A_{dn}$ has all eigenvalues within a constant factor $\left(\frac{\sqrt{d}+1}{\sqrt{d}-1}\right)^2$ of each other and is hence a good approximation to the identity.

Batson, Spielman, and Srivastava use two "barrier" potential functions to guide their choice of $\pi(i)$ and $s_{\pi(i)}$ and ensure steady progress. Specifically, for $u, l \in \mathbb{R}$, and A a symmetric matrix with eigenvalues $\lambda_1, \ldots, \lambda_n$, they define

$$\Phi^u(A) \stackrel{\text{def}}{=} \mathrm{Tr}(uI - A)^{-1} = \sum_i \frac{1}{u - \lambda_1} \quad \text{(Upper potential)}$$

$$\Phi_l(A) \stackrel{\text{def}}{=} \mathrm{Tr}(A - uI)^{-1} = \sum_i \frac{1}{\lambda_1 - l} \quad \text{(Lower potential)}$$

When $l \cdot I_n \prec A \prec u \cdot I_n$, small values of these potentials indicate that the eigenvalues of A do not cluster near u or l. This turns out to be a sufficient induction hypothesis to sustain the following iterative process:

(1) Begin by setting the lower barrier l, to $-n$ and the upper barrier, u to n. It can be checked that both potentials are bounded by 1. (2) At each step, increase the upper barrier u by a fixed constant δ_u and the lower barrier l by another fixed constant $\delta_l < \delta_u$. It can then be shown that as long as the potentials remain bounded, there must exist at every time t a choice of a vector $v_{\pi(i)}$ and a weight $s_{\pi(i)}$ so that the addition of $s_{\pi(i)} v_{\pi(i)} v_{\pi(i)}^T$ to A_{t-1} and the increments $l \to l + \delta_l$ and $u \to u + \delta_u$ *do not increase either potential* and keep all the eigenvalues $\lambda_i(A_t)$ between the barriers. Iterating the above process ensures steady growth of all the eigenvalues and yields Theorem 8.

3.5. Some extensions

In a recent work, de Carli Silva et al.[15] extended spectral sparsification to the sums of positive semidefinite matrices that have arbitrary rank. They proved the following theorem.

THEOREM 9. *Let $B_1, \ldots, B_m$ be symmetric (or Hermitian) positive semidefinite $n \times n$ matrices and $B = \sum_{i=1}^{m} B_i$. Then for any $\epsilon \in (0, 1)$, there exist nonnegative $s_1, \ldots, s_m$, at most $O(n/\epsilon^2)$ of which are nonzero, such that*

$$B \preccurlyeq \sum_i s_i B_i \preccurlyeq (1+\epsilon) \cdot B.$$

Moreover, $\{s_1, \ldots, s_m\}$ can be computed in $O(mn^3/\epsilon^2)$ time.

Another extension is subgraph spectral sparsification,[22] in which one is given a union of two graphs G and W and an integer κ, and asked to find a κ-edge graph W_κ such that $G + W_\kappa$ is a good spectral sparsifier of $G + W$. When combined with the best-known construction of low-stretch spanning trees,[1] this provides nearly optimal *ultra-sparsifiers*.

THEOREM 10 (KOLLA, MAKARYCHEV, SABERI, TENG). *For each positive integer κ, every n-vertex graph has an $O(\sqrt{(n/\kappa)\log n \log n \log n})$-spectral approximation with at most $(n - 1 + \kappa)$-edges.*

Ultra-sparsifiers have so few edges that they have a large number of vertices of degree 1 or 2. They are a key component of the algorithms for solving linear equations described in Section 4.1.

4. ALGORITHMIC APPLICATIONS

4.1. Numerical algorithms: Laplacian systems

One of the most fundamental computational problems is that of solving a system of linear equations. One is given an $n \times n$ matrix A and an n-dimensional vector b, and is asked to find a vector x such that $Ax = b$. It is well known that every linear system can be solved by the classic Gaussian elimination method in polynomial time. However, Gaussian elimination usually takes super-linear or even super-quadratic time in the number of non-zero entries of A, making its use impractical for large matrices.

In many real-world applications, the matrix A is sparse and one only requires an approximate solution to the linear system. For example, given a precision parameter ϵ, we may be asked to produce an $\tilde{x}$ such that $\|A\tilde{x} - b\|_2 \leq \epsilon \|b\|_2$. For sparse positive semi-definite linear systems the fastest general purpose algorithm is the Conjugate Gradient (CG). It essentially solves $Ax = b$ by multiplying a sequence of vectors by A. As the multiplication of a vector by A takes time proportional to the number of non-zero entries in A, CG can run quickly when A is sparse The number of matrix-vector products performed by CG depends on the *condition number* $\kappa(A)$ of A, the ratio of its largest eigenvalue to its smallest eigenvalue. It is well-known in numerical analysis[19] that it is sufficient to compute $O(\sqrt{\kappa(A)}\log(1/\epsilon))$ matrix-vector products to find a solution of accuracy ϵ.

Preconditioning is the strategy of finding a relatively easily invertible matrix B which is σ–spectrally similar to A, and solving the related system $B^{-1}Ax = B^{-1}b$. In each iteration, the preconditioned CG algorithm solves a linear system in B and performs a matrix-vector product in A, and only $O(\sqrt{\kappa(B^{-1}A)}\log(\epsilon^{-1})) = O(\sigma\log(\epsilon^{-1}))$ iterations are required. If it is easy to solve systems of linear equations in B, then the cost of each iteration is small and this algorithm will run quickly.

In 1990, Vaidya brought graph theory into the picture. Using preconditioners consisting of a maximum spanning tree of a graph plus a small number of carefully chosen edges, Vaidya obtained an $O(m^{1.75}\log(1/\epsilon))$-time algorithm for solving linear systems in Laplacian matrices with m non-zero entries. The exponent was still too large to be practical, but the idea was powerful. Spielman and Teng[33] were able to enhance Vaidya's approach with spectral sparsifiers and low-stretch spanning trees to obtain the first nearly linear time algorithm for solving Laplacian linear systems.

THEOREM 11 (SPIELMAN-TENG). *Linear systems in a graph Laplacian L_G can be solved to precision ϵ in time $O(m\log^{O(1)} n \log(1/\epsilon))$.*

In spite of its strong asymptotic behavior, the large exponent on the log factor makes this algorithm slow in practice. The Spielman-Srivastava sparsification algorithm offered no improvement—effective resistances do give the ideal probabilities with which to sample edges for a sparsifier, but computing them requires solving yet more Laplacian linear systems.

Koutis et al.[24] removed this dependency problem by using low-stretch trees to compute less aggressive sampling probabilities which are strictly greater than those suggested by effective resistances. This can be done more quickly, and along with some other elegant ideas and fast data structures, is sufficient to yield a Laplacian linear system solver which runs in time

$$O(m \log n \log\log^2 n \log(1/\epsilon)).$$

4.2. Fast sparsification algorithms

The algorithms from Section 3 certified the existence of good sparsifiers, but run quite slowly in practice. Those techniques have been significantly refined, and now there are three major ways to produce sparsifiers quickly.

First, the bottleneck in sampling via effective resistances is approximating the effective resistances themselves. The Laplacian system solver of Koutis, Miller, and Peng described above can be used to calculate those resistances, which can then be used to sample the graph. The best analysis is given by Koutis et al.[23] who give an $O(m \log n \log\log n \log(1/\epsilon))$ time algorithm for generating $((1+\epsilon), O(\log^3 n/\epsilon^2))$-spectral sparsifiers.

Second, the decomposition-and-sampling algorithm of Spielman and Teng[34] can be sped up by improving the local clusterings used to create a decomposition. In the local clustering problem, one is given a vertex and cluster size as input, and one tries to find a cluster of low conductance near that vertex of size at most the target size, in time proportional to the target cluster size. Faster algorithms for local clustering have been developed by Andersen et al.[5] and by Andersen and Peres.[6]

Third, unions of random spanning trees of a graph G can make good cut-sparsifiers: the union of two random spanning trees is $(\log n)$-cut similar to G,[20] while the union of $O(\log^2 n/\epsilon^2)$ random spanning trees, reweighed proportionally to effective resistance, is $(1+\epsilon)$-cut similar to G.[18] Although it remains to be seen if the union of a small number of random spanning trees can produce a spectral sparsifier, Kapralov and Panigrahy showed that one can build a $(1+\epsilon)$-spectral sparsifier of a graph from the union of spanners of $O(\log^4 n/\epsilon^4)$ random subgraphs of G.[21]

4.3. Network algorithms

Fast algorithms for spectral sparsification and Laplacian systems provide a set of powerful tools for network analysis. In particular, they lead to nearly-linear time algorithms for the following basic graph problems:

Approximate Fiedler Vectors[33]: *Input*: $G=(V,E,w)$ and $\epsilon>0$. *Output*: an ϵ-approximation of the second smallest eigenvalue, $\lambda_2(L_G)$, (also known as the Fiedler value) of L_G, along with a vector v orthogonal to the all 1s vector such that $v^T L_G v \le (1+\epsilon)\lambda_2(L_G)$.

Electrical Flows[13, 32, 33]: *Input*: $G=(V,E,w)$ where weights are resistances, s, $t \in V$, and $\epsilon>0$. *Output*: an ϵ-approximation of the electrical flows over all edges when 1 unit of flow is injected into s and extracted from t.

Effective Resistance Approximation[32]: *Input*: $G=(V,E,w)$ where weights are resistances and $\epsilon>0$. *Output*: a data structure for computing an ϵ-approximation of the effective resistance of any pair of vertices in G. Data Structure: $O(n \log n/\epsilon^2)$ space, and $O(\log/\epsilon^2 n)$ query time, and $O(m \log^2 n \log\log^2 n/\epsilon^2)$ preprocessing time.

Learning from labeled data on a graph[35]: *Input* a strongly connected (aperiodic) directed graph $G=(V,E)$ and a labeling function y, where y assigns a label from a label set $Y=\{1,-1\}$ to each vertex of a subset $S \subset V$ and 0 to vertices in $V-S$, and a parameter μ. *Output* the function $f: V \to \mathbb{R}$ that minimizes

$$\Omega(f)+\mu\|f-y\|^2,$$

where

$$\Omega(f)=\sum_{(u,v)\in E}\pi(u)p(u,v)\left(f(u)\pi(u)^{-1/2}-f(v)\pi(v)^{-1/2}\right),$$

$$p(u,v)=\begin{cases}1/(\text{out}-\text{degree of } u) & \text{for } (u,v)\in E,\\ 0 & \text{otherwise,}\end{cases}$$

and π is the stationary distribution of the random walk on the graph with transition probability function p.

Cover Time of Random Walk[16]: *Input*: $G=(V,E)$, *Output*: a constant approximation of the cover time for random walks.

The algorithm for Electrical Flows led to a breakthrough for the following fundamental combinatorial optimization problem, for which the best previously known algorithm ran in time $\tilde{O}(mn^{1/2}/\epsilon)$, $\tilde{O}(mn^{2/3}\log\epsilon^{-1})$ and $\tilde{O}(m^{3/2}\log\epsilon^{-1})$.

Maximum Flows and Minimum Cuts[13]: *Input:* $G=(V,E,w)$ where w are capacities, s, $t \in V$, and $\epsilon>0$, *Output*: an ϵ-approximation of s-t maximum flow and minimum cut. *Algorithm*: $\tilde{O}(mn^{1/3}\cdot \text{poly}(1/\epsilon))$ time.

Spectral graph sparsification also played a role in understanding other network phenomena. For example, Chierichetti et al.[12] discovered a connection between rumor spreading in a network and the spectral sparsification procedure of Spielman and Teng,[34] and applied this connection to bound the speed of rumor spreading that arises in social networks.

5. OPEN QUESTION

The most important open question about spectral sparsification is whether one can design a nearly linear time algorithm that computes (σ, d)-spectral sparsifiers for any constants σ and d. The algorithms based on Theorem 7 are polynomial time, but slow. All of the nearly-linear time algorithms of which we are aware produce sparsifiers with d logarithmic in n.

6. CONCLUSION

Spectral sparsification has proved to be a remarkably useful tool in algorithm design, linear algebra, combinatorial

optimization, machine learning, and network analysis. Theorem 8 has already been applied many times within pure mathematics (see, e.g., Naor[28]). We hope this body of work will encourage more exchange of ideas between numerical analysis, pure mathematics and theoretical computer science, and inspire and enable the development of faster algorithms and novel analyses of network phenomena. C

References

1. Abraham, I., Neiman, O. Using petal-decompositions to build a low stretch spanning tree. In (2012).
2. Achlioptas, D., Mcsherry, F. Fast computation of low-rank matrix approximations. , 2 (2007), 9.
3. Alon, N., Milman, V.D. λ_1, isoperimetric inequalities for graphs, and superconcentrators. , 1 (1985), 73–88.
4. Althöfer, I., Das, G., Dobkin, D., Joseph, D., Soares, J. On sparse spanners of weighted graphs. (1993), 81–100, 10.1007/BF02189308.
5. Andersen, R., Chung, F., Lang, K. Local graph partitioning using pagerank vectors. In (2006), 475–486.
6. Andersen, R., Yuval, P. Finding sparse cuts locally using evolving sets. In (2009), ACM, 235–244.
7. Batson, J.D., Spielman, D.A., Srivastava, N. Twice-Ramanujan sparsifiers. 6 (2012), 1704 –1721.
8. Benczúr, A.A., Karger, D.R. Approximating s-t minimum cuts in $O(n^2)$ time. In (1996), 47–55.
9. Bollobás, B. The isoperimetric number of random regular graphs. , 3 (May 1988), 241–244.
10. Chandra, A.K., Raghavan, P., Ruzzo, W.L., Smolensky, R., Tiwari, P. The electrical resistance of a graph captures its commute and cover times. In (1989), ACM, New York, NY, USA, 574–586.
11. Cheeger, J. A lower bound for smallest eigenvalue of Laplacian. In (1970), Princeton University Press, 195–199.
12. Chierichetti, F., Lattanzi, S., Panconesi, A. Rumour spreading and graph conductance. In (2010), 1657–1663.
13. Christiano, P., Kelner, J.A., Madry, A., Spielman, D.A., Teng, S.H. Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs. In (2011), 273–282.
14. Chung, F.R.K. Spectral graph theory. CBMS Regional Conference Series in Mathematics, American Mathematical Society, 1997.
15. de Carli Silva, M.K., Harvey, N.J.A., Sato, C.M. Sparse sums of positive semidefinite matrices. (2011), abs/1107.0088.
16. Ding, J., Lee, J.R., Peres, Y. Cover times, blanket times, and majorizing measures. In (2011), 61–70.
17. Friedman, J. A Proof of Alon's Second Eigenvalue Conjecture and Related Problems. Number 910 in Memoirs of the American Mathematical Society. American Mathematical Society, 2008.
18. Fung, W.S., Hariharan, R., Harvey, N.J.A., Panigrahi, D. A general framework for graph sparsification. In (2011), 71–80.
19. Golub, G.H., Van Loan, C.F. Matrix Computations, 2nd edn, Johns Hopkins University Press, 1989.
20. Goyal, N., Rademacher, L., Vempala, S. Expanders via random spanning trees. In (2009), 576–585.
21. Kapralov, M., Panigrahy, R. Spectral sparsification via random spav nners. In (2012), 393–398.
22. Kolla, A., Makarychev, Y., Saberi, A., Teng, S.H. Subgraph sparsification and nearly optimal ultrasparsifiers. In (2010), 57–66.
23. Koutis, I., Levin, A., Peng, R. Improved spectral sparsification and numerical algorithms for SDD matrices. In (2012), 266–277.
24. Koutis, I., Miller, G.L., Peng, R. A nearly-m log n time solver for SDD linear systems. In (2011), 590–598.
25. Leiserson, C. Fat-trees: universal networks for hardware-efficient supercomputing. , 10 (October1985), 892–901.
26. Lubotzky, A., Phillips, R., Sarnak, P. Ramanujan graphs. , 3 (1988), 261–277.
27. Margulis, G.A. Explicit group theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators. , 1 (July 1988), 39–46.
28. Naor, A. Sparse quadratic forms and their geometric applications. (2011), 1033.
29. Peleg, D., Ullman, J.D. An optimal synchronizer for the hypercube. , 4 (1989), 740–747.
30. Rudelson, M. Random vectors in the isotropic position. , 1 (1999), 60–72.
31. Sinclair, A., Jerrum, M. Approximate counting, uniform generation and rapidly mixing Markov chains. 1 (July 1989), 93–133.
32. Spielman, D.A., Srivastava, N. Graph sparsification by effective resistances. 6 (2011), 1913–1926.
33. Spielman, D.A., Teng, S.H. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. (2008), abs/cs/0607105.
34. Spielman, D.A., Teng, S.H. Spectral sparsification of graphs. , 4 (2011), 981–1025.
35. Zhou, D., Huang, J., Scholkopf, B. Learning from labeled and unlabeled data on a directed graph. In (2005), 1041–1048.
36. Zhu, X., Ghahramani, Z., Lafferty, J.D. Semi-supervised learning using Gaussian fields and harmonic functions. In (2003).

Joshua Batson, Mathematics, MIT.

Daniel A. Spielman, Computer Science & Applied Mathematics, Yale University.

Nikhil Srivastava, Microsoft Research, Bangalore.

Shang-Hua Teng, Computer Science, USC.