# COMS 4232 Advanced Algorithms: Spectral Sparsification

Team: Manuel Paez, Frederick Cunningham, Adrien Ben Natan

May 10th 2023

## 1 Introduction

Spectral sparsification of a graph is the procedure of removing edges from it such that the graph spectral properties are preserved. Although there are other notions of graph similarity, spectral metrics generalize beyond combinatorial ones thanks to the connections underlying Cheegar's inequality. In addition, spectral methods come packaged with optimized algorithms for common spectral tasks that can be applied to sparsification. To provide a mathematical definition of spectral sparsification, we consider the general case of the undirected and weighted graph $G = (V, E, w)$ with $n$ vertices and $m$ edges such that $V = [n]$ is the set of nodes, $E \subset \{(u, v) \mid u, v \in V\}$ is the edge set, and weight function $w : V \times V \to \mathbb{R}^+$. The Laplacian matrix of $G$ can be thought of as the adjacency matrix subtracted from the degree matrix and is defined to be an $n \times n$ matrix $L$ where

$$L_G(u, v) = \begin{cases} -w(u, v) & \text{if } u \sim v \\ deg(u) & \text{if } u = v \\ 0 & \text{otherwise} \end{cases}$$

where $deg(u) = \sum_{u \sim v} w(u, v)$. With this, it can be shown that the Laplacian Matrix may be defined by its quadratic form

$$Q_G(x) = x^T L_G x = \sum_{u \sim v} w(u, v)(x_u - x_v)^2 \geq 0 \quad \text{for any } x \in \mathbb{R}^n$$

The quadratic form of a graph is connected to its spectral properties and its cut boundaries. Considering $x \in \mathbb{R}^n$, such that, $\|x\| = 1$, the maximum and minimum quadratic forms of a matrix are the largest and smallest eigenvalues of the matrix respectively. When components of $x$ represent 1-0 membership of a set $S \subset V$ ($x_i = \mathbb{1}[i \in S]$), then, $Q_G(x) = 2|\partial S|$ where $\partial S$ represents the set of edges that connect a vertex in $S$ with a vertex in $V - S$. Note the conductance of a cut $S$ of $G$ is,

$$\Phi_G(S) \triangleq \frac{|\partial S|}{\min\{\text{Vol}(S), \text{Vol}(V - S)\}}$$

where $\text{Vol}(S) = \sum_{i \in S} deg(i)$ Since the conductance of a cut can be understood as the ratio of the number of edges that cross the boundary of a cut of $V$ to the number of edges contained in a cut of $V$, we can also define the conductance of a graph $G$ by

$$\Phi_G \triangleq \min_{\emptyset \neq S \subset V} \Phi_G(S).$$

Even without a proof, it can be seen that there is a connection between the number of edges connecting a min cut and the eigenvalues of the Laplacian. Consider the normalized Laplacian

$$\hat{L}_G \triangleq D^{-1/2} L_G D^{1/2}$$

where $D$ is the diagonal matrix representing the degree of each vertex. It turns out that the first non-zero eigenvalue of the normalized Laplacian represents the conductance of a graph such if the magnitude of this eigenvalue is zero, it implies the graph contains two disconnected components and if the magnitude of this eigenvalue is one, it implies the conductance of any cut is high. This is formalized in Cheegar's inequality which states that

$$2/\Phi_G \leq \mu_2 \leq 2\Phi_G$$

where $\mu_2$ is the second smallest eigenvalue of $\hat{L}_G$. With this, we can introduce an intuitive definition of spectral similarity by defining a notion of similarity that preserves the eigenvalues of the normalized Laplacian of a graph $G$ and its sparsifier $\widetilde{G}$. With an understanding of the relationship between the quadratic form and the spectral properties of a matrix, it is natural to define that $\widetilde{G}$ is a $\sigma$-spectral approximation of $G$ if for all $x \in \mathbb{R}^n$

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

We may condense this definition by saying the sub-graph $\widetilde{G}$ is a $\sigma$-spectral approximation of $G$ if

$$\frac{1}{\sigma} L_{\widetilde{G}} \preceq L_G \preceq \sigma L_{\widetilde{G}},$$

where the notation, $A \preceq B$, implies that for $A, B \in \mathbb{R}^{n \times n}$, for all $x \in \mathbb{R}^n$, $x^T A x \leq x^T B x$. To further understand this definition, consider a graph that has two connected components that are very dense, but there are only a few edges connecting the two components (shown in fig.1). Intuitively, removing

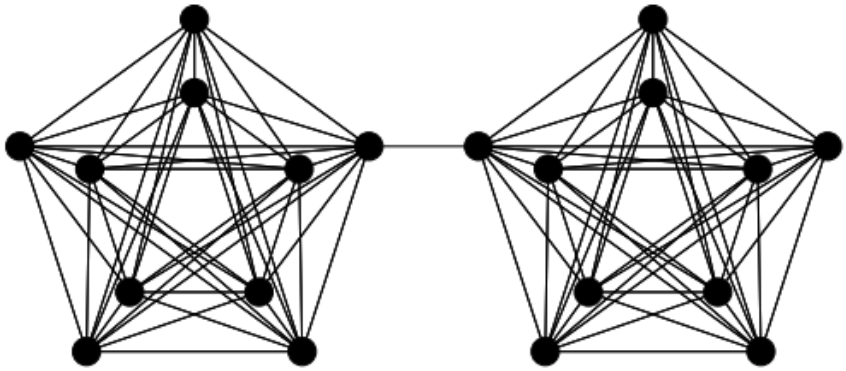

Figure 1: Two highly connected graphs connected by a single edge taken from [ST11].

an edge from the boundary of the two equal-volume connected components will have a greater impact on similarity than removing an edge from one of the connected components. Through the relationship between conductance and spectrum defined above, we can see that removing certain boundary edges such

as the edge connecting two dense components will also impact the spectral properties of the sparsifier. This example not only helps to explain the definition of spectral sparsification, but also suggests a method of sparsifying graphs by removing edges that approximately maintain the conductance of the original graph.

## 2 Preliminaries

Throughout this paper we will assume the graph construction described in the in Introduction. For any matrix, let $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ be the maximum and minimum eigenvalues of $A$. The condition number of matrix $A$ is defined by $\lambda_{\max}(A)/\lambda_{\min}(A)$. For any two matrices $A$ and $B$, we write $A \preceq B$ to represent that $B - A$ is positive semidefinite and $A \prec B$ to represent that $B - A$ is positive definite. For any two matrices $A$ and $B$ of equal dimensions, let $A \cdot B = tr(A^T B)$. For any function $f$, we write $\tilde{O}(f) \triangleq O(f \cdot log^{O(1)} f)$. For matrices $A$ and $B$, we write $A \approx_\epsilon B$ if $(1-\epsilon) \cdot A \preceq B \preceq (1+\epsilon)A$.

## 3 Nearly Linear Time Sparsifiers

The first spectral sparsification procedure, taken from Spielman and Teng [ST11], introduces a simple extension of the ideas presented in the introduction. First, partition a graph into subgraphs of high conductance. Second, randomly sample edges to remove from these graphs from the high conductance subgraphs but preserve the boundaries. Lastly, if there are too many edges remaining in the boundary, consider a new graph where each subgraph is a node connected to other subgraph nodes with boundary edges from the original graph. Then recursively run the algorithm on this new graph to remove edges from the boundary. As this rather simple idea leads to a complex implementation that approximates the NP-hard exact algorithm, we will focus on the case of sparsifying an unweighted graph and we will omit many details from the proof of correctness. According to the authors, their main result is showing the existence of good spectral sparsifiers with a nearly linear number of edges and that their discovery is feasible in better than polynomial time.

### 3.1 Notation and Tools

For graphs $G$ and $H$, we will use the notation $G + H$ to refer to the graph with Laplacian $L_G + L_H$. For graphs $G$ and $H$, it is immediate that $G + H \preccurlyeq \widetilde{G} + \widetilde{H}$. Note that when a subgraph is constructed, its node numeric identities will be preserved allowing the sum of subgraphs to yield results about the original graph. We define the boundary of a vertex set $S$ with respect to another vertex set $B$ to be the set of edges from $S$ into $B - S$ denoted $\partial_B(S)$ or $E(S, B - S)$. We define the conductance of a set $S$ in the subgraph induced by $B \subseteq V$ to be,

$$\Phi_B^G(S) \triangleq \frac{|\partial_B(S)|}{\min\{\mathrm{Vol}(S), \mathrm{Vol}(B-S)\}}$$

and we define

$$\Phi_B^G \triangleq \min_{S \subset B} \Phi_B^G(S).$$

We also define that $\Phi_B^G(\emptyset) = 1$ and for $|B| = 1, \Phi_B^G = 1$. We will assume that if a graph has high conductance, then it may be sparsified by a random sampling procedure such that $E[\widetilde{A}] = A$ where $\widetilde{A}$

and $A$ are the adjacency matrices of the sparsified graph and the original graph respectively. While we include the specific sampling algorithm here we do not analyze its specifics as there are a variety that can be chosen. This algorithm in particular is tuned such that there is zero probability of removing all edges from a node. The internal parameter $\gamma$ controls the number of edges expected in the samples graph. Note that if $\gamma = n \triangleq |V|$, then the sampled graph will be exactly the same as the original. The algorithm below is taken directly from [ST11]. Theorem 1 ensures that this procedure applied to graph of high

**Algorithm 2** Algorithm to randomly sample from graph of high conductance

1: $\widetilde{G} = \text{Sample}(G, \epsilon, p, \lambda)$
2: Set $k = \max\{\log_2(3/p), \log_2(n)\}$
3: Set $\gamma = \left(\frac{12k}{\epsilon\lambda}\right)^2$
4: **for** every edge $(i, j)$ in $G$ **do**
5: Set the probability $p_{i,j} = \min\{1, \frac{\gamma}{\min\{d_i, d_j\}}\}$
6: Place an edge of weight $1/p_{i,j}$ between vertices $(i, j)$ into $\widetilde{G}$
7: **end for**

conductance ensures a good approximation.

**Theorem 1.** *Let $\epsilon, p \in (0, 1/2)$ and let $G = (V, E)$ be an unweighted graph whose smallest non-zero normalized Laplacian eigenvalue is at least $\lambda$. Let $S \subset G$ and let $F$ be the edges in $G(S)$. Suppose $(S, \widetilde{F}) = \text{Sample}((S, F), \epsilon, p, \lambda)$ and let $\widetilde{G} = (V, \widetilde{F} \cup (E - F))$. Then with probability at least 1-p,*

1. *$\widetilde{G}$ is a $(1 + \epsilon)-$approximation of $G$, and*
2. *The number of edges in $\widetilde{F}$ is at most*

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

Lets define a decomposition of $G$ into vertex sets $(A_1, ..., A_k)$ for some $k$ as a $\phi-$decomposition if $\Phi^G_{A_i} \geq \phi$ for all $i$. We define the boundary of a decomposition as the set of edges between different vertex sets in the partition denoted,

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

Considering the conductance of a subgraph, the boundary of a decomposition gives us a decomposition where each subgraph has conductance at least $\phi$. We also define a decomposition as a $\lambda-$spectral decomposition if the smallest non zero normalized Laplacian eigenvalue of $G(A_i)$ is at least $\lambda$, for all $i$. Using Cheegar's inequality we have that every $\phi-$decomposition is a $(\phi^2/2)-$spectral decomposition.

### 3.2 Existence of Sparsifiers

In this section, we will show the existence of sparsifiers for the procedure for generating sparse graphs with our desired properties. Theorem 3 is central to the existence of sparsifiers and in its proof there will be ideas on how to construct sparsifiers. The proof of Theorem 3 follows quickly from lemma 2 which makes a statement connecting the size of a sparse cut to its conductance.

**Lemma 2.** *Let $G = (V, E)$ be a graph and let $\phi \leq 1$. Let $B \subseteq V$ and let $S \subset B$ be a set maximizing* $\mathrm{Vol}(S)$ *among those satisfying:*

1. $\mathrm{Vol}(S) \leq \mathrm{Vol}(B)/2$, *and*
2. $\Phi_B^G(S) \leq \phi$

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

$$\Phi_{B-S}^G \geq \phi \left( \frac{1-3\alpha}{1-\alpha} \right)$$

*Proof.* Consider $B$ a vertex set of some graph $G$. Let $S$ maximize $\mathrm{Vol}(S)$ and satisfy condition 1 and 2 of Lemma 2. $S$ is the largest volume vertex set in $B$ of bounded conductance. Let $\beta = \frac{1-3\alpha}{1-\alpha}$. Suppose contrarily that $\Phi_{B-S}^G < \phi\beta$. Suppose $R$ is the set that minimizes $\Phi_{B-S}^G$. Then $\Phi_{B-S}^G(R) < \phi$ and $\mathrm{Vol}(R) \leq \mathrm{Vol}(B-S)/2$. Let $T = R \cup S$. Note that

$$\begin{aligned} |E(T, B-T)| &\leq |E(S, B-S)| + |E(R, B-S-R)| && (1) \\ &< \phi \mathrm{Vol}(S) + \phi\beta \mathrm{Vol}(R) && (2) \end{aligned}$$

Consider 2 cases

1. Suppose $\mathrm{Vol}(T) \leq \mathrm{Vol}(B)/2$. This implies that $\mathrm{Vol}(S) \leq \mathrm{Vol}(B)/2$ because of $T$ has strictly more connections than $S$. Thus

$$\begin{aligned} |E(T, B-T)| &< \phi(\mathrm{Vol}(S) + \beta \mathrm{Vol}(R)) \\ &< \phi(\mathrm{Vol}(S) + \mathrm{Vol}(R)) = \phi \mathrm{Vol}(T) \\ \Rightarrow \frac{|E(T, B-T)|}{\mathrm{Vol}(T)} &= \Phi_B^G(T) < \phi \end{aligned}$$

   However, this is a contradiction because we stated that $S$ was the largest set that satisfied these properties and $T$ is strictly larger than $S$.

2. Suppose that $\mathrm{Vol}(T) > \mathrm{Vol}(B)/2$. Note that

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

   because $\mathrm{Vol}(S) = \alpha \mathrm{Vol}(B)$. From the definition of conductance, $\mathrm{Vol}(S) \leq (1/2)\mathrm{Vol}(B)$ and that $\mathrm{Vol}(R) \leq (1/2)(\mathrm{Vol}(B-S)$ This implies that

$$\begin{aligned} \mathrm{Vol}(B-T) &= \mathrm{Vol}(B) - \mathrm{Vol}(T) \\ &\geq \mathrm{Vol}(B)(1-\alpha)/2 \end{aligned}$$

Now consider the conductance of $T$,

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

Thus, we have an upper bound on the conductance of $T$ within the subgraph induced by $B$

$$\begin{aligned}\Phi_B^G(T) &= \frac{|E(T, B-T)|}{\min\{\text{Vol}(T), \text{Vol}(B-T)\}} \\ &= \frac{|E(T, B-T)|}{\text{Vol}(B-T)} \text{ by case 2 assumption} \\ &\leq \frac{\phi \text{Vol}(B)(\alpha + \beta(1-\alpha)/2)}{\text{Vol}(B)(1-\alpha)/2} \\ &= \phi\end{aligned}$$

$T$ contradicts the constructive assumption of the conditions of $S$.

Therefore, Lemma 2 holds. □

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

*Proof.* To show Theorem 3, we will show that Algorithm 3 yields the decomposition that is claimed to exist. First we see that this procedure must terminate because the recursive call is always passed a strictly smaller set than the calling function was passed and $\Phi_B^G = 1$ when $|B| = 1$ and $\phi < 1$. Let $(A_1, ..., A_k)$ be the output of idealDecomp($V$). Note that set $A_i$ can be returned when $\Phi_{A_i}^G \geq \phi$ or when $\Phi_{A_i}^G \geq \phi\frac{1-3\alpha}{1-\alpha}$. So in either case $\Phi_{A_i}^G \geq \phi/3$ for each $i$. Now consider the depth of the recursion. In the worst case the $\text{Vol}(S) \leq \text{Vol}(S)/4$ at every iteration because that maximizes the size of the set passed to idealDecomp. Therefore, at each iteration the volume of the set passed recursively will be at least 3/4 times the volume of set passed to the caller. Therefore, because the volume of a set cannot be less than 1 we have that $1 \leq \text{Vol}(V)(3/4)^d$ where $d$ is the worst case depth of the recursion. Therefore,

$$d \leq \log_{4/3} \text{Vol}(V).$$

Now supplement this finding with the fact that every time a set is split into $S$ and $B - S$ we add edges to $\partial(A_1, ..., A_k)$. However, since $S$ satisfies Lemma 2, we know that $\Phi_B^G(S) \leq \phi$. Therefore, every time we split a set we are adding at most $|E|\phi$ edges to $\partial(A_1, ..., A_k)$ because $|E|$ is always larger than $\min\{\text{Vol}(S), \text{Vol}(S)\}$. Therefore,

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

Since $\text{Vol}(V) = 2m$, we have that $\phi/3 \leq \left(2 \log_{4/3} 2m\right)^{-1}$ implies

$$\phi \leq \left(6 \log_{4/3} 2m\right)^{-1}$$

□

**Algorithm 3** Algorithm to partition graph with desired properties

1: Set $\phi = \left(2\log_{4/3}\text{Vol}(V)\right)^{-1}$
2: Note, the algorithm is called initially with $B = V$
3: idealDecomp$(B, \phi)$
4: If $\Phi_B^G \geq \phi$ then return $B$. Otherwise proceed.
5: Let $S$ be the subset of $B$ maximizing Vol$(B)$ satisfying the conditions of lemma 2
6: If Vol$(S) \leq$ Vol$(B)/4$, return the decomposition ($B - S$, idealDecomp$(S, \phi)$
7: else, return the decomposition (idealDecomp$(B - S, \phi)$, idealDecomp$(S, \phi)$)

Since edges are bounded by $n^2$ for $n = |V|$, then Theorem 3 shows the decomposition with minimum conductance of $(6\log_{4/3} 2m)^{-1}$. Since the conductance is $\Omega(1/\log(n))$, then by Cheegar's inequality, the second lowest non-zero eigenvalue is $\Omega(1/\log^2(n))$. Therefore, we can apply Theorem 1 to each set of the partition to ensure that the number of edges in these partitions are $\widetilde{O}(n)$. All that remains is to ensure that the edges in the boundary of the decomposition are also $\widetilde{O}(n)$. This can be accomplished by collapsing all the nodes in each partition into a single node of a new graph and recursively partitioning this new graph with Theorem 3. Everytime we use Theorem 3, the number of edges in the resulting boundary is divided by 2. Therefore, we will re-curse to a depth logarithmic in $|E|$. The crux of this proof is the procedure of partitioning a graph into connected components of "high" conductance. Due to Lemma 2, we have a set of conditions that will guarantee that the complement of the set that satisfies them has high enough conductance. Those necessary conditions are numbered 1 and 2 in Lemma 3 as well as the size limit on the set produced. Our idealized decomposition algorithm extracts a set of high conductance if all the conditions are met and re-runs the algorithm on the rest. If the size condition is not met, it simply re-runs the algorithm on both the set and its complement.

### 3.3 Approximation

Unfortunately, the previous section skips over one important detail. It is not specified exactly how we can find a set that satisfies the conditions of Lemma 2. If this were clear we would have a perfectly functional algorithm for spectral sparsification. Line 5 of Algorithm 3 is where this breaks down. Finding such a set is an NP-hard problem, and therefore must be approximated. We will not go into the details, but there exists an algorithm that can generate these sets approximately in linear time:

$$D = \text{ApproxCut}(G, \phi, p),$$

where $G$ is the graph to find the cut within, $\phi \in (0, 1)$ is the conductance, and $p \in (0, 1)$ is the probability of failiure where success is defined as the output set $D \subset V$ as meeting conditions.

1. Vol$(D) \geq (1/29)$Vol$(V)$ or
2. there exists a set $W \supset V - D$ for which $\Phi_W^G \geq f_2(\phi)$ where $f_2(\phi) \triangleq \frac{c_2\phi^2}{\log^4 m}$ for some constant $c_2$

Note that condition 2 is essentially the same thing as saying that instead of the complement of $D$ having high conductance, there exists a subset of the complement of $D$ that has high conductance. Also $D$ is guaranteed to satisfy,

1. $\text{Vol}(D) \leq (23/25)\text{Vol}(V)$ and

2. for $D \neq \emptyset$, $\Phi_G(D) \leq \phi$

The expected runtime of ApproxCut is $O(\phi^{-4} m \log^9 m \log(1/p))$. Now using this approximation of line 5 of Algorithm 3, we introduce Algorithm 4 which sparsifies an unweighted graph using approximate sparsest cuts and our random sampling algorithm from the first section. In simple english, this algo-

**Algorithm 4** Approximate algorithm to sparsify an unweighted graph

1: $\widetilde{G} = \text{UnwtedSparsify}(G, \epsilon, p)$
2: If $\text{Vol}(V) \leq c_3 \epsilon^{-2} n \log^3 0(n/p)$, return $G$
3: Set $\phi = (2 \log_{29/28} \text{Vol}(V))^{-1}$, $\hat{p} = p/6n \log_2 n$, and $\hat{\epsilon} = \frac{\epsilon (\ln 2)^2}{(1+2\log_{29/28} n)(2 \log n)}$
4: Set $(\widetilde{G}_1, ..., \widetilde{G}_k) = \text{PartitionAndSample}(G, \phi, \hat{\epsilon}, \hat{p})$.
5: Let $V_1, ..., V_k$ be the vertex sets of $\widetilde{G}_1, ..., \widetilde{G}_k$ respectively, and let $G_0$ be the graph with vertex set $V$ and edge set $\partial(V_1, ..., V_k)$.
6: Set $\widetilde{G} = \sum_{i=0}^{k} \widetilde{G}_i$
7: $(\widetilde{G}_1, ..., \widetilde{G}_k) = \text{PartitionAndSample}(G = (V, E), \phi, \hat{\epsilon}, \hat{p})$
8: Set $\lambda = f_2(\phi)^2/2$
9: Set $D = \text{ApproxCut}(G, \phi, \hat{p})$
10: If $D = \emptyset$ return $G_1 = \text{Sample}(G, \hat{\epsilon}, \hat{p}, \lambda)$
11: Else, if $\text{Vol}(V) \leq (1/29)\text{Vol}(V)$
12: Set $\widetilde{G}_1 = \text{Sample}(G(V - D), \hat{\epsilon}, \hat{p}, \lambda)$
13: Return ($\widetilde{G}_1$, PartitionAndSample($G(D), \phi, \hat{\epsilon}, \hat{p}$))
14: Else
15: Set $\widetilde{H}_1, ..., \widetilde{H}_k = \text{PartitionAndSample}(G(V - D), \phi, \hat{\epsilon}, \hat{p})$
16: Set $\widetilde{I}_1, ..., \widetilde{I}_j = \text{PartitionAndSample}(G(D), \phi, \hat{\epsilon}, \hat{p})$
17: Return $(\widetilde{H}_1, ..., \widetilde{H}_k, \widetilde{I}_1, ..., \widetilde{I}_j)$

rithm uses a subroutine, PartitionAndSample that uses ApproxCut to find high conductance vertex sets. Whenever it does find one, it instantly calls Sample on it. When ApproxCut does not directly yield a high conductance cut, (when the set it returns is large) PartitionAndSample is recursively called on the cut and its complement. The caller of PartitionAndSample, UnwtedSparsify, sparsifies a given partition and then acts recursively on the remaining edges in boundary set of the partition. The specifics of this algorithm are included more for completeness than understanding and we do not analyze it fully. In addition, the constants in Algorithm 4 make it entirely impractical for reasonable sized graphs. However, it does produce a $(1 + \epsilon)-$approximation of $G$ with at most $c_4 \epsilon^{-2} n \log^{31}(n/p)$ edges, for some constant $c_4$ with probability $p$. It achieves an expected runtime of $O(m \log(1/p) \log^{15} n)$

# 4 Graph Sparsification by Effective Resistances

## 4.1 Introduction

In 2008, Spielman and Srivastava [SS08] introduce a nearly-linear time algorithm which produces sparsifiers with $O(nlogn/\epsilon^2)$ edges by using random sampling over the edges of $G$. For an edge $e \in E$, they introduce a notion of resistance which is equal to $1/w_e$, and a notion of effective resistance $R_e$ which is the potential difference induced across it when a unit current is injected at one end and extracted at the

other. It is known to be proportional to the commute time of an edge, which is the expected time it takes for a natural random walk starting at one end to travel to the other end and back.

### 4.2 Main algorithm

We denote $q$ to be the sample size.

**Algorithm 6** Effective Resistances Sparsifier Algorithm

```
Initially, set G̃ = (V, Ẽ = ∅, w̃)
for i in range(q) do
    Pick a random edge e from G with probability p_e proportional to w_e R_e
    if e is not already in Ẽ then
        Add e to Ẽ with weight w̃_e = w_e / q p_e
    else
        Set w̃_e = w̃_e + w_e / q p_e
    end if
end for
```

If $q = 4C^2 nlogn/\epsilon^2$, where $C$ is a constant from a concentration theorem by Rudelson & Vershynin [RV06], then $\tilde{G}$ is a sparsifier with probability at least 1/2. This is proven using Markov inequality on

$$\tilde{L} = B^T W^{1/2} S W^{1/2} B$$

where $S, W \in R^{m \times m}$ are such that $\forall e \in E$, $S(e,e) = \dfrac{\tilde{w}_e}{w_e}$, $W(e,e) = w_e$, and $B$ is the *signed edge-vertex incidence matrix* of $G$ defined as

$$B(e,v) = \begin{cases} 1 & \text{if } v \text{ is } e\text{'s head} \\ \text{-1} & \text{if } v \text{ is } e\text{'s tail} \\ 0 & \text{otherwise} \end{cases}$$

after orienting the edges arbitrarily if $G$ is undirected. Once the effective resistances have been computed, the main algorithm runs in time $O(nlogn/\epsilon^2)$.

### 4.3 Computing the effective resistances

The key part of this algorithm is approximating the effective resistances $\{R_e\}_{e \in E}$ needed for sampling in nearly-linear time. Since we are comparing the effective resistances in the main algorithm, it is enough to approximate them all up to the same common factor. The authors use the *Moore-Penrose Pseudoinverse* $L_G^+$ of $L_G$ defined as

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

where $\lambda_1, ..., \lambda_{n-1}$ are the non-zero eigenvalues of $L_G$ and $u_1, ..., u_{n-1}$ a corresponding set of orthonormal eigenvectors, to build the set

$$\{W^{1/2} B L_G^+ \chi_v\}_{v \in V}$$

where $\chi_u$ is the elementary vector with coordinate 1 in position $u$. They prove that for $u, v \in V$, the effective resistance $R_e$ of the edge $e$ between $u$ and $v$ is the distance between $W^{1/2}BL_G^+\chi_u$ and $W^{1/2}BL_G^+\chi_v$. Using the Johnson-Linderstrauss Lemma [JL84], these vectors are projected onto $\mathbb{R}^{O(logn)}$ to accelerate the distance computation. The resulting set is

$$\{QW^{1/2}BL_G^+\chi_v\}_{v\in V}$$

where $Q \in \mathbb{R}^{k\times n}$ is a random $\pm 1/\sqrt{k}$ matrix, and $k$ is given by the Achlioptas Lemma [Ach01] in order to maintain $\epsilon$-approximate distances.

**Algorithm 8** Computing the Effective Resistances

1: Compute such a random $Q \in \mathbb{R}^{k\times n}$ for $k = 24logn/\epsilon^2$
2: Compute $Y = QW^{1/2}B$
3: **for** $i$ in range($k$) **do**
4: Compute $z_i$ where $z_iL = y_i$ where $y_i$ is the $i$-th row of $Y$ by calling the Spielman-Teng solver [ST08]
5: **end for**
6: Return the matrix $Z$ whose $i$-th row is $z_i$

The matrix $Z$ is equal to $QW^{1/2}BL_G^+$, and its columns can be subtracted 2 by 2 to find distances between the vectors of the set and the effective resistances. Computing $Y$ takes time $2m\times 24logn/\epsilon^2+m = \tilde{O}(m/\epsilon^2)$ since B has $2m$ entries and $W^{1/2}$ is diagonal. Each call to the Spielman-Teng solver takes $\tilde{O}(m)$ time, leading to the construction of $Z$ in time $\tilde{O}(m/\epsilon^2)$. Thus, for each pair $u, v \in V$, we can compute the effective resistance $R_e$ of the edge between $u$ and $v$ in time $O(logn)$.

# 5 Twice-Ramanujan Sparsifiers

## 5.1 Introduction

Ramanujan graphs are connected $d$-regular graphs $H$, for which

$$\forall \lambda \in Sp(L_H), \lambda \neq 0 \implies \lambda \in [d - 2\sqrt{d-1}, d + 2\sqrt{d-1}]$$

Building a Ramanujan graph on $n$ vertices and multiplying the weight of every edge by $n/(d - 2\sqrt{d-1})$ give us a graph which $\kappa$-approximates the complete graph for

$$\kappa = \frac{d + 2\sqrt{d-1}}{d - 2\sqrt{d-1}}$$

## 5.2 The Main Result

In their paper [BSS14] from 2009, Batson, Spielman and Srivastava show that every graph can be approximated at least this well for $d > 1$ by a graph with only twice as many edges as a Ramanujan graph. In other words, they show this very theorem:

**Theorem 4.** *For every $d > 1$, every undirected weighted graph $G = (V, E, w)$ on $n$ vertices contains a*

*weighted subgraph $H = (V, F, \tilde{w})$ with $\lceil d(n-1) \rceil$ edges (i.e., average degree at most $2d$) that satisfies:*

$$x^T L_G x \leq x^T L_H x \leq \left( \frac{d+1+2\sqrt{d}}{d+1-2\sqrt{d}} \right) \cdot x^T L_G x \quad \forall x \in \mathbb{R}^V$$

The main result which leads to Theorem 4 is the following:

**Theorem 5.** *Suppose $d > 1$, and $v_1, v_2, ..., v_m$ are vectors in $\mathbb{R}^n$ with*

$$\sum_{i \leq m} v_i v_i^T = \boldsymbol{id}_{R^n}$$

*Then there exists scalars $s_i \geq 0$ with $|\{i : s_i \neq 0\}| \leq dn$ so that*

$$I_n \preceq \sum_{i \leq m} s_i v_i v_i^T \preceq \left( \frac{d+1+2\sqrt{d}}{d+1-2\sqrt{d}} \right) \cdot \boldsymbol{id}_{R^n}$$

We define

$$V_{n \times m} = (L_G^+)^{1/2} B^T W^{1/2}$$

where $B$ and $L_G^+$ are respectively the *signed edge-vertex incidence* and the *Moore-Penrose Pseudoinverse* matrices defined in the previous section and $W \in \mathbb{R}^{m \times m}$ is the diagonal matrix where $\forall e \in E$, $W(e,e) = w_e$. Because $L_G L_G^+ = L_G^+ L_G = \mathbf{id}_{im(L_G)}$, $V_{n \times m}$ yields columns $\{v_i\}_{i \in [m]}$ such that

$$\begin{aligned} \sum_{i \leq m} v_i v_i^T &= VV^T \\ &= (L_G^+)^{1/2} B^T W B (L_G^+)^{1/2} \\ &= (L_G^+)^{1/2} L_G (L_G^+)^{1/2} \\ &= \mathbf{id}_{im(L_G)} \end{aligned}$$

Using Theorem 5, we obtain the diagonal matrix $S_{m \times m}$ such that $\forall i \in [m]$, $S(i,i) = s_i$ and set

$$L_{\tilde{G}} = B^T W^{1/2} S W^{1/2} B$$

which is the Laplacian of the subgraph $\tilde{G}$ of $G$ with weights $\{\tilde{w}_i = w_i s_i\}_{i \in E}$ ; $\tilde{G}$ has at most $d(n-1)$ edges since at most that many of the $s_i$ are nonzero. Using the Courant-Fischer Theorem, we can prove

$$1 \leq \frac{x^T L_{\tilde{G}} x}{x^T L_G x} \leq \left( \frac{d+1+2\sqrt{d}}{d+1-2\sqrt{d}} \right) \quad \forall x \perp \mathbf{1}$$

as desired. The correct set of weights $\{s_i \in R^+ \mid i \in [m]\}$ is picked using an algorithm described below, which gives a constructive proof to Theorem 5 and thus concludes the proof to Theorem 4.

### 5.3 The Algorithm

The first observation to be made is that for $A \in \mathbb{R}^{n\times n}$ and $v \in \mathbb{R}^n$, the eigenvalues of $A + vv^T$ interlace those of $A$. The eigenvalues of $A$ can be seen as charged particles on a slope, resting on chargeless barriers because of gravity. Adding $vv^T$ to $A$ corresponds to adding a charge $< v, u_j >^2$ on the barrier corresponding to the $j$-th eigenvalue, thus repelling the initial eigenvalue forward on the slope. The idea is that by adding sufficiently enough $vv^T$ for $v$ picked at random, all of the eigenvalues move up at the same pace and the condition number $\dfrac{\lambda_{max}(A)}{\lambda_{min}(A)}$ is bounded.

The algorithm of Theorem 4 takes its inspiration from this model and builds $A^{(q)}$ from $A^{(q-1)}$ iteratively by picking $i \in [m]$, $s_i > 0$ and adding $s_i v_i v_i^T$ while keeping all of the eigenvalues between two barriers. To that end, the authors introduce two functions that describe how far the eigenvalues of $A$ are from the lower and upper barriers $l, u \in \mathbb{R}$:

$$\Phi^u(A) = Tr(uI - A)^{-1} \quad \text{(Upper potential)}$$
$$\Phi_l(A) = Tr(A - lI)^{-1} \quad \text{(Lower potential)}$$

It is very important that the interval between $\lambda_{min}(A)$ and $\lambda_{max}(A)$ does not increase as we iterate, and for this reason the Both Barriers Lemma is introduced. It proves that for a given $A$ and values $\delta_L > 0$, $\delta_U > 0$, there always exists an $i \in [m]$ and $s_i > 0$ such that we can shift the lower and upper barriers respectively by $\delta_L$ and $\delta_U$ forward and maintain the upper and lower potentials, i.e.

$$\Phi^{u+\delta_U}(A + s_i v_i v_i^T) \leq \Phi^u(A)$$
$$\Phi_{l+\delta_L}(A + s_i v_i v_i^T) \leq \Phi_l(A)$$

The index $i$ and value $s_i$ are picked using special matrices referred to as $U_{A^{(q)}}$ and $L_{A^{(q)}}$; these matrices result from a computation involving $A$, the potentials and the barriers.

**Algorithm 10** Twice-Ramanujan Sparsifiers Algorithm

1: Compute the columns $\{v_i\}_{i\in[m]}$ of $V_{n\times m} = (L_G^+)^{1/2} B^T W^{1/2}$
2: Initially, set $A^{(0)} = 0$, the barriers $u = u_0$ and $l = l_0$, and the potentials

$$\Phi_0^u(A^{(0)}) = \epsilon_U \quad \text{and} \quad \Phi_{l_0}(A^{(0)}) = \epsilon_L$$

3: **for** $q$ in range$(dn)$ **do**
4: Set $u = u + \delta_U$ and $l = l + \delta_L$
5: Set $A^{(q+1)} = A^{(q)} + s_i v_i v_i^T$, where $v_i$ and $s_i$ are picked so that the potentials don't increase.
6: **end for**

After $Q = dn$ steps, we have

$$\frac{\lambda_{max}(A^{(Q)})}{\lambda_{min}(A^{(Q)})} \leq \frac{u_0 + dn\delta_u}{l_0 + dn\delta_L}$$

By choosing the right values $u_0, l_0, \delta_U, \delta_L, \epsilon_U$, and $\epsilon_L$, we get

$$\frac{\lambda_{max}(A^{(Q)})}{\lambda_{min}(A^{(Q)})} \leq \frac{d+1+2\sqrt{d}}{d+1-2\sqrt{d}}$$

as desired by Theorem 5, which leads to the sparsifier $\tilde{G}$.

Computing the vectors $v_i$ is done in time $O(n^2m)$. For each iteration of the algorithm, computing the matrices $U_{A^{(q)}}$ and $L_{A^{(q)}}$ needed to pick the next $v_i$ to add can be performed in time $O(n^3)$. Finally, we need to compute $U_{A^{(q)}}(v_i)$ and $L_{A^{(q)}}(v_i)$ for each edge $v_i$, which can be done in time $O(n^2m)$. As we run for $dn$ iterations, the total time for the algorithm is $O(dn^3m)$.

# 6 Fast Construction of Linear-size Spectral Sparsifers

## 6.1 Constructing a Randomize Algorithm for Fast Spectral Sparsification

It is known that constructing a spectral sparsifier for graphs is a special case of sparsifying the sum of rank-1 PSD matrices. With this, [LS18] goal was to create a near-linear time algorithm such that for any vectors $v_1, ...v_m$ with $\sum_{i=1}^m v_i v_i^T = I$, it finds scalars $\{s_i\}_{i=1}^m$ satisfying

$$|\{s_i : s_i \neq 0\}| = O\Big(\frac{qn}{\epsilon^2}\Big)$$

such that

$$(1-\epsilon)\cdot I \preceq \sum_{i=1}^m s_i v_i v_i^T \preceq (1+\epsilon)\cdot I$$

For inspiration, [LS18] builds upon [BSS14] by improving on a randomized version of their algorithm, which is shown below. As described previously, [BSS14] uses two barrier values $u_j$ and $l_j$ and defines two

**Algorithm 11** Randomized [BSS14] Algorithm for spectral sparsification

$j = 0$
$l_0 = -(2n)/\epsilon, u_0 = (2n)/\epsilon, A_0 = 0$
**while** $u_j - l_j < 8n/\epsilon$ **do**
    Let $t = tr(u_jI - A_j)^{-1} + tr(A_j = l_jI)^{-1}$
    Sample a vector $v_i$ with probability

$$p_i = \Big(v_i^T(u_jI - A_j)^{-1} + v_i^T(A_j - l_jI)^{-1}v_i\Big)/t$$

    $A_{j+1} = A_j + \frac{\epsilon}{t}\cdot\frac{1}{p_i}\cdot v_i v_i^T$
    $u_{j+1} = u_j + \frac{\epsilon}{t\cdot(1-\epsilon)}, l_{j+1} = l_j + \frac{\epsilon}{t\cdot(1+\epsilon)}$
    $j = j+1$
**end while**
**return** $A_j$

potential functions $\Phi^u(A) \triangleq tr(uI - A)^{-1}$ and $\Phi_l(A) \triangleq tr(A - lI)^{-1}$ to determine if some eigenvalue of $A$ is close to the barriers $u$ and $l$. During an iteration $j$, their Algorithm always find a vector $\{v_i\}_{i=1}^m$ and updates $u_j, l_j$ such that the invariant $l_jI \prec A_j \prec u_jI$ holds in each iteration. After $T = \Theta(n/\epsilon^2)$

iterations, they would find a matrix $A_T \approx_{O(\epsilon)} I$. However, areas of improvement can be made to the randomized [BSS14] algorithm.

1. We need a fast algorithm to approximate the probabilities $\{p_i\}_{i=1}^m$ used to choose vectors
2. We need to use the computed $\{p_i\}_{i=1}^m$ to choose multiple vectors in each phase to reduce the total number of phases

With this, [LS18] improves on both points for their algorithm by adopting the potential function

$$\Phi_{u,l}(A) \triangleq tr(uI - A)^{-q} + tr(A - lI)^{-q}$$

to maintain that the eigenvalues of $A$ do not get close to the barriers $u$ and $l$ in order to approximate the probabilities $\{p_i\}_{i=1}^m$ efficiently (since constant $q$ is large) and by showing that the potential function still works as before as long as the sampling probability satisfies

$$p_i \geq C \cdot \frac{v_i^T(uI - A)^{-1}v_i + v_i^T(A - lI)^{-1}v_i}{\sum_{j=1}^m \left(v_j^T(uI - A)^{-1}v_j + v_j^T(A - lI)^{-1}v_j\right)}$$

for some constant $C > 0$. With this, the algorithm only needs to recompute $\{p_i\}_{i=1}^m$ after every $\Omega(n^{1-1/q})$ iterations. As the algorithm chooses $\Theta(n/\epsilon^2)$ vectors in total, the algorithm only recomputes the probabilities $\Theta(\frac{n^{1/q}}{\epsilon^2})$ times. In essence, [LS18] is constructed such that it uses the same probability $\{p_i\}_{i=1}^m$ to choose multiple vectors before the algorithm increases the barrier values $u_j, l_j$ for each iteration. The algorithm is shown below:

**Algorithm 12** [LS18]'s algorithm for fast-construction of spectral sparsifiers

**Require:** $\epsilon \leq 1/120, q \geq 10$

$j = 0$
$l_0 = -(2n)^{1/q}, u_0 = (2n)^{1/q}, A_0 = 0$
**while** $u_j - l_j < 4 \cdot (2n)^{1/q}$ **do**
    $W_j = 0$
    Compute $R_i(A_j, u_j, l_j)$ for all vectors $v_i$
    Sample $N_j$ vectors independently with replacement, where every $v_i$ is chosen with probability proportional to $R_i(A_j, u_j, l_j)$. For every sampled $v$, add $\epsilon/q \cdot (R_i(A_j, u_j, l_j))^{-1} \cdot vv^T$ to $W_j$
    $A_{j+1} = A_j + W_j$
    $u_{j+1} = u_j + \Delta_{l,j}, l_{j+1} = l_j + \Delta_{l,j}$
    $j = j + 1$
**end while**

To analyze the algorithm further, given the initial values $j$, $l_0, u_0, A_0, \epsilon$ and $q$ set in the formal framework of the algorithm, the algorithm proceeds in iterations where in each iteration $j$, it computes the relative effective resistance $R_i(A_j, u_j, l_J)$ of vectors $\{v_i\}_{i=1}^m$ where

$$R_i(A_j, u_j, l_j) \triangleq v_i^T(u_jI - A_j)^{-1}v_i + v_i^T(A_j - l_jI)^{-1}v_i$$

After this, the algorithm samples $N_j$ vectors independently with replacement, where vector $v_i$ is chosen with probability proportional to $R_i(A, u, l)$. The number of samples $N_j$ in iteration $j$ is defined by

$$N_j \triangleq \frac{1}{n^{2/q}}\Big(\sum_{i=1}^{m} R_i(A_j, u_j, l_j)\min\{\lambda_{\min}(u_j I - A_J), \lambda_{\min}(A_j - l_j I)\}\Big)$$

After each iteration $j$ the algorithm updates $u_j$ with $\Delta_{u,j}$, $l_j$ with $\Delta_{l,j}$, and $A_j$ with $W$ where $W$ is defined to be

$$W = \sum_{\text{sampled vector } v_i} \frac{\epsilon}{q} \cdot \frac{1}{R_i(A_j, u_j, l_j)} \cdot v_i v_i^T$$

Using the expected value of $W$ defined as $E[W] = \frac{\epsilon}{q} \cdot \frac{N_j}{R_i(A_j, u_j, l_j)} \cdot I$, the algorithm increases the barrier values $u_j$ and $l_j$ by $\Delta u, j$ and $\Delta l, j$ respectively where

$$\Delta_{u,j} \triangleq (1+3\epsilon) \cdot \frac{\epsilon}{q} \cdot \frac{N_j}{\sum_{i=1}^{m} R_i(A_j, u_j, l_j)} \quad \text{and} \quad \Delta_{l,j} \triangleq (1-3\epsilon) \cdot \frac{\epsilon}{q} \cdot \frac{N_j}{\sum_{i=1}^{m} R_i(A_j, u_j, l_j)}$$

The choice of $N_j$, $\Delta_{u,j}$ and $\Delta_{l,j}$ ensures that $0 \preceq W \preceq \frac{1}{2} \cdot (u_j I - A_j)$ holds with high probability and that the invariant (shown below) holds for any phase $j$

$$l_j I \prec A_j \prec u_j I$$

In essence, by the last iteration $T$, $l_T I \prec A_T \prec u_T I$ will hold, resulting in the a matrix $A_T$ that is linear-sized where $A_T \approx_{O(\epsilon)} I$.

### 6.2 Run-time of LS18's Fast Spectral Sparsifier algorithm

We want to show that the algorithm's run-time for producing a $(1+O(\epsilon))$-spectral sparsifier is near-linear; specificially we want to show the algorithm's runtime being $\tilde{O}\big(\frac{qm}{\epsilon^2} \cdot n^{\omega-1+3/q}\big)$. For this construction, we use $v_1, ..., v_N$ to denote the $N$ sampled vectors and we denote the reweighted vectors for any $1 \leq i \leq N$ as $w_i \triangleq \sqrt{\frac{\epsilon}{q \cdot R_i(A,u,l)}} \cdot v_i$. With this, we let matrix $W$ be $W \triangleq \sum_{i=1}^{N} w_i w_i^T$.

**Lemma 6.** *Assume that the number of sampled vectors satisfies* $N < \frac{2}{n^{2/q}}\Big(\sum_{i=1}^{m} R_i(A, u, l) \cdot \lambda(uI - A)\Big)$. *It holds that*

$$\mathbb{E}[W] = \mathbb{E}\Big[\sum_{1}^{N} w_i w_i^T\Big] = \frac{\epsilon}{q} \cdot \frac{N}{\sum_{i=1}^{m} R_i(A, u, l)} \cdot I$$

*and*

$$Pr[0 \preceq W \preceq \frac{1}{2} \cdot (uI - A)] \geq 1 - \frac{\epsilon^2}{100qn}$$

**Lemma 7.** *The output matrix $A_k$ has a condition number at most $1 + O(\epsilon)$*

**Lemma 8.** *The following statements hold true:*

1. *The algorithm finishes in $\frac{10qn^{3/q}}{\epsilon^2}$ phases with probability at least 4/5*
2. *The algorithm chooses at most $\frac{10qn}{\epsilon^2}$ vectors with probability at least 4/5*

Although we will not prove Lemma 7, its importance allows us to show that the Algorithm does produce a linear-sized $(1+O(\epsilon))$-spectral sparsifier by showing that the condition number of $A_k$ is small. This is done by noticing that the barrier gap $u_j - l_j$ increases after every iteration $j$ for point 1 and that the barrier gap $\Delta_{u,j} - \Delta l, j$ increases on average for every vector chosen in iteration $j$ for point 2. Now, we want to construct an assumption for approximating the required quantities quickly for constructing graph sparsifiers. For this, let us assume the following: let $L$ and $\tilde{L}$ be the Laplacian matrices of graph $G$ and its subgraph after re-weighting, let $A = L^{-1/2}\tilde{L}L^{-1/2}$, and assume that

$$(l + |l|\eta) \cdot I \prec A \prec (1-\eta)u \cdot I$$

holds for some $0 < \eta < 1$. Thus, the next two lemmas hold under this assumption:

**Lemma 9.** *Let $A = \sum_{i=1}^{m} v_i v_i^T$ and suppose that $A$ satisfies the assumption. Then, we can compute $\{r_i\}_{i=1}^m$ and $\{t_i\}_{i=1}^m$ in $\tilde{O}(\frac{m}{\epsilon^2\eta})$ such that*

$$(1-\epsilon)r_i \leq v_i^T(ul - A)^{-1}v_i \leq (1+\epsilon)r_i \quad \text{and} \quad (1-\epsilon)t_i \leq v_i^T(A - lI)^{-1}v_i \leq (1+\epsilon)t_i$$

**Lemma 10.** *We can compute values $\alpha, \beta$ in $\tilde{O}(\frac{m}{\eta\epsilon^3})$ time such that*

$$(1-\epsilon)\alpha \leq \lambda_{\min}(uI - A) \leq (1+\epsilon)\alpha \quad \text{and} \quad (1-\epsilon)\beta \leq \lambda_{\min}(A - lI) \leq (1+\epsilon)\beta$$

Using the assumption and the lemmas that follow from it, we can prove the following theorem:

**Theorem 11.** *Let $G = (V, E, w)$ be an undirected and weighted graph with $n$ vertices and $m$ edges, and let $q \geq 10, 0 < \epsilon \leq 120$ be constants. Then there is an algorithm that outputs a $(1+\epsilon)$-spectral sparsifier of $G$ with $O(qn/\epsilon^2)$ edges. The algorithm runs in $\tilde{O}\big(\frac{q \cdot m \cdot n^{5/q}}{\epsilon^{4+4/q}}\big)$ time.*

*Proof.* We want to construct our proof by firstly proving that the Algorithm does output a linear-size spectral sparsifer. WLOG, consider the graph $G = (V, E, 2)$ defined earlier. Suppose for every edge $e = (u, v) \in E$ where $u, v \in V$, we define the vector $b_e \in \mathbb{R}^n$ to be $b_e(x) = 1$ if $x = u, b_e(x) = -1$ if $x = v$, and 0 otherwise. The Laplacian matrix of graph $G$ would be $L = \sum_{e \in E} b_e b_e^T$. If we let $v_e = L^{-1/2}b_e$, we can show that constructing a spectral sparsifier of $G$ is equivalent to sparsifying the matrix $\sum_{e \in E} v_e v_e^T$. Lemma 7 and 8 allow us to say that $A_k$ is a linear-sized spectral sparsifier, partly as it allows us to prove that $A_k$ is a $(1+O(\epsilon)$ approximation of I. To analyze the runtime of the algorithm, we know from Lemma 6 that with probability at least 9/10, all matrices picked in $k = \frac{10qn^{3/q}}{\epsilon^2}$ phases satisfy $W_j \preceq \frac{1}{2}(u_j I - A_j)$. If this even occurs, then it holds for any phase $j$ that

$$\mathbb{E}\Big[\Phi_{u_j,l_j}(A_j) | \forall j : W_j \preceq \frac{1}{2}\big(u_j I - A_J\big)\Big] \leq 2$$

Thus, by Markov's inequality, for all iterations $j, \Phi_{u_j,l_j}(A_j) = O(qn^{3/q}/\epsilon^2)$ holds with high probability. Notice that for eigenvalues $\lambda_1, ..., \lambda_n$ of matrix $A$, it holds that

$$(u - \lambda_j)^{-q} \leq \sum_{i=1}^{n} (u - \lambda_i)^{-q} < \Phi_{u,l}(A)$$

for any $1 \leq j \leq n$. This implies that $\lambda_j < u - (\Phi_{u,l}(A))^{-1/q}$ and $\lambda_j > l - (\Phi_{u,l}(A))^{-1/q}$. Thus, we have

$$\Big(l_j + O\Big(\Big(\frac{\epsilon^2}{qn^{3/4}}\Big)^{1/q}\Big)\Big)I \preceq A_j \preceq \Big(u_j - O\Big(\Big(\frac{\epsilon^2}{qn^{3/4}}\Big)^{1/q}\Big)I$$

Since $u_j$ and $l_j$ are of the order $O(n^{1/q})$, we can set $\eta = O\big((\epsilon/n)^{2/q}\big)$ to get

$$\big(l_j + |l_j|\eta\big)I \preceq A_j \preceq \big(1-\eta\big)u_j I$$

We can apply Lemma 9 and 10 to compute all required quantities in each phase for a constant approximation in time.

$$\tilde{O}\Big(\frac{m}{\epsilon^2} \cdot \eta\Big) = \tilde{O}\Big(\frac{m \cdot n^{2/q}}{\epsilon^{2+2/q}}\Big)$$

Since Lemma 8 says the algorithm finishes in $\frac{10qn^{3/q}}{\epsilon^2}$ phases with probability $4/5$, the total run-time of the algorithm is

$$\tilde{O}\Big(\frac{m \cdot n^{2/q}}{\epsilon^{2+2/q}}\Big) \cdot \frac{10qn^{3/q}}{\epsilon^2} = \tilde{O}\Big(\frac{q \cdot m \cdot n^{5/q}}{\epsilon^{4+4/q}}\Big)$$

which completes our proof □

Now we want to show the goal of [LS18]: to find a near-linear time algorithm for outputting scalars $\{s_i\}_{i=1}^m$ with $|\{s_i : s_i \neq 0\}| = O(qn/\epsilon^2)$ such that

$$(1-\epsilon) \cdot I \preceq \sum_{i=1}^{m} s_i v_i v_i^T \preceq (1+\epsilon) \cdot I$$

Using Lemma 6, Theorem 11, and fast matrix multiplication, all required quantities can be computed for each iterature in $\tilde{O}(m \cdot n^{\omega-1})$ time (where $\omega$ is a matrix-multiplication constant). Thus, the total run-time is

$$\tilde{O}\Big(\frac{qm}{\epsilon^2} \cdot n^{\omega-1+3/q}\Big) \tag{3}$$

which proves our run-time claim for the algorithm and that the algorithm runs in near-linear time.

# References


[Ach01] Dimitris Achlioptas. Database-friendly random projections. In *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '01, page 274–281, New York, NY, USA, 2001. Association for Computing Machinery.

[BSS14] Joshua Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-Ramanujan Sparsifiers. *SIAM Review*, 56(2):315–334, 2014.

[JL84] William Johnson and Joram Lindenstrauss. Extensions of lipschitz maps into a hilbert space. *Contemporary Mathematics*, 26:189–206, 01 1984.

[LS18] Yin Tat Lee and He Sun. Constructing Linear-Sized Spectral Sparsification in Almost-Linear Time. *SIAM Journal on Computing*, 47(6):2315–2336, 2018.

[RV06] Mark Rudelson and Roman Vershynin. Sampling from large matrices: an approach through geometric functional analysis, 2006.

[SS08] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, page 563–568, New York, NY, USA, 2008. Association for Computing Machinery.

[ST08] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems, 2008.

[ST11] Daniel A Spielman and Shang-Hua Teng. Spectral Sparsification of Graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.