Graph Sparsification via Effective Resistance and Some Empirical Results

Yue Yang Kenrick Fernandes Rami Melhem

Abstract

Graph sparsification is the approximation of an arbitrary graph by a sparse graph. Given a graph G, sparsification outputs a graph H that has the same set of vertices as G but with fewer edges and is close to G by certain metrics. A good graph sparsification scheme can be a remarkably useful tool in graph-related probelems since a sparser graph is cheaper to both store and compute with. We focus on one particular sparsification method—sparsification via effective resistance introduced by Spielman and Srivastava and test its implementation on both synthetic and natural graphs. Results from our experiments demonstrate expected computational behaviors specified by the algorithm. We believe that this sparsification approach can be further employed in the design of faster graph algorithms.

1 Introduction

In this paper, we primarily study the spectral sparsification introduced by Spielman and Srivastava that samples edges of a graph according to their effective resistances. This approach is an example of *importance sampling*. We will explain the intuition and theoretical framework of their algorithm and demonstrate a more efficient way to compute the approximations. In addition, we present an implementation of the algorithm and give our empirical results and evaluations of the sparsification scheme.

1.1 Background and Motivation

Graph models are ubiquitous for computational analysis in various practical applications. Many problems involving network analysis rely on the mathematical model of graphs and their related properties. Therefore, it is of great interest to explore approaches to both store and compute graph models more efficiently. One useful method to achieve this is graph sparsification. Graph sparsification generates a sparse approximation of the relatively dense original graph while preserving the fundamental properties of the original graph such as cut sizes and clusters. Generally, we consider a graph to be dense if its number of edges is close to $O(n^2)$ relative to the number of its vertices, while a sparse graph usually has a number of edges that are reasonably proportional to its number of vertices. Sparsification is done by sampling edges from the original graph and outputs a sparse subgraph on the same set of vertices. This is a desired outcome because the performance of most graph algorithms depend on both the orders of vertices and edges of the graph. Therefore, a good sparse approximation guarantees faster computation while preserving the correctness of the results relatively well.

One may argue that sparsification in most cases will not be necessary since most natural graphs are very sparse to begin with. However, graphs from real world applications can be quite large and extensive in size. A social network graph, such as a data model representing the shared social circles on Google+ has 107,614 vertices and 13,673,453 edges [8]. A graph model of such size takes a significant amount of space to store. Therefore, a sparsification method that reduces the magnitude of edges provides great storage benefits.

2 Sparsification by Random Sampling

The graph sparsification scheme designed by Spielman and Srivastava relies on graph spectral theory, which studies the properties of a graph in relations to the graph's associated matrices and their eigenvalues and eigenvectors. Matrices closely related to a graph usually include the *adjacency* and *Laplacian* matrix. The main idea of spectral sparsification is to sample the edges of the graph with an assigned probability. This probability is proportional to how critical each edge is in maintaining the properties of the graph such as communities and connectivities. This notion of $importance$ is defined to be the $effective\ resistance$ of each edge.

2.1 Preliminaries

In order to understand Spielman and Srivastava's theoretical framework, we first clarify certain definitions and notations. We specify a weighted graph by a 3-tuple, $G = (V, E, w)$, with vertex set $V = \{1, \dots, n\}$, edge set $E \subseteq \{(u, v)|u, v \in V\}$, and weights $w_{(u,v)} > 0$ for each $(u, v) \in E$. All graphs will be undirected, weighted, and connected unless otherwise stated. In addition, when measuring the similarity between two graphs, we will always assume that they have the same set of vertices.

If we orient the edges of G arbitrarily–that is, we assign an arbitrary direction to each edge in the graph with n vertices and m edges, then we can obtain the *signed edge-vertex incidence matrix* $B_{m\times n}$ where

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

If we donote the row vectors of B by $\{b_e\}_{e\in E}$, then $b_e^T = (\chi_v - \chi_u)$ where χ_u is the elementary unit vector with a coordinate 1 in position u. Finally, we define the diagonal matrix $W_{m\times m}$ with entries $W(e, e) = w_e$.

The **Laplacian matrix** of a graph is a symmetric matrix defined as $L_G := D_G$ - A_G, where D_G is the diagonal *degree matrix* of G and A is G's *adjacency matrix*. With the arbitrarily oriented edges and their incidence vectors, we can write L_G as

$$\begin{aligned} L_G &= \sum_{e\in E} w_e b_e b_e^T \\ &= B^T W B. \end{aligned} \tag{1}$$

Given a weighted graph G, we define its **Laplacian quadratic form** to be the function Q_G from $\mathbb{R}^V$ to $\mathbb{R}$ given by

$$\begin{aligned} Q_G(x) &= x^T L_G x \\ &= \sum_{(u,v)\in E} w_{(u,v)}(x(u) - x(v))^2 \end{aligned}$$

where $x \in \mathbb{R}^V$ is a function vector over the vertices in G. Since the quadratic form is always nonnegative, it follows that L_G is *positive semidefinite*. This means that L_G has exactly one eigenvalue λ_0 that is equal to 0. Since L_G is *singular*, we can only specify its *Moore-Penrose Pseudoinverse* (or simply *Pseudoinverse*), which can be written as

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

where $\lambda_1, \ldots, \lambda_{n-1}$ are the nonzero eigenvalues of L_G and $u_1, \ldots, u_{n-1}$ are the corresponding eigenvectors. The *Pseudoinverse* of a matrix is essentially a more generalized form of the *inverse* of a matrix. In the case where a matrix A is not invertible, its *Pseudoinverse* is the "closest" answer to the system of linear equations $Ax = b$ [3]. This will be a useful notion in the calculation of *effective resistances*. Finally, the definition of L_G^+ also states that

$$L_G^+ = L_G^+ L_G L_G^+.$$

The kernel (or nullspace) of L_G is denoted by $\ker(L_G)$. Then we also have $\ker(L_G)$ $=\ker(L^+)$ and that

$$LL^+ = L^+L = \sum_{i=1}^{n-1} u_i u_i^T,$$

which is simply the projection onto the span of the nonzero eigenvectors of L_G (which are also the eigenvectors of L_G^+). This means that, for our approximation purpose, if we only consider L_G on its *image* $\mathrm{im}(L_G)$ where L_G has full rank and is invertible ($L_G^{-1} = L_G^+$), then $L_G L_G^+ = L_G^+ L_G$ is equal to the identity matrix on $\mathrm{im}(L)$.

2.2 Cut and Spectral Similarity

Spielman and Srivastava's notion of spectral similarity was first inspired by Benczur and Karger's cut similarity [4]. As part of an effort to develop fast algorithms for the minimum cut and maximum flow problems, Benczur and Karger defined that 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, or a 'cut' of the vertices into two parts can be specified by identifying the subset of vertices in one part. For a weighted graph $G = (V, w)$ and a subset of vertices $S \subset V$, we define

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

We say that G and $\tilde{G}$ are *σ-cut similar* if

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

for all $S \subset V$. Benczur and Karger also showed that every graph is cut-similar to a graph with average degree $O(\log n)$ [4].

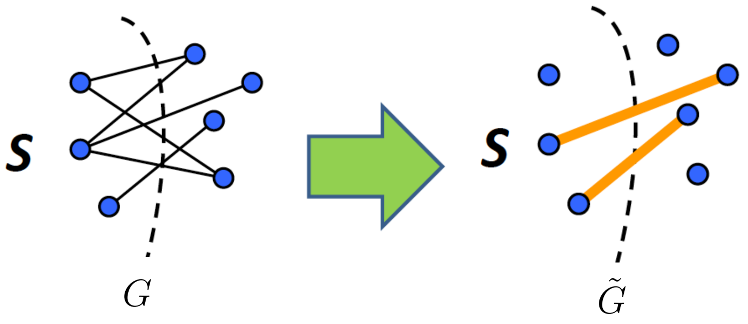


Figure 1: Graph $\tilde{G}$ with rescaled edge weights is a *σ-cut approximation* of graph G for a cut S [11].

It follows from the notion of Laplacian quadratic form that, if we let x be the characteristic vector of a cut S (1 inside S and 0 outside), then we have

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

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

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

Then we can infer that spectral-similarity is *stronger* than cut-similarity. In other words, cut-similarity can be viewed as a special case of spectral similarity in which we only consider vectors x that take discrete values in $\{0, 1\}$ [12]. For two symmetric matrices A and B in $\mathbb{R}^{n\times n}$, we write $A \preceq B$ to indicate that

$$x^T Ax \leq x^T Bx, \text{for all } x \in \mathbb{R}^n.$$

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

$$B/\sigma \preceq A \preceq \sigma \cdot B.$$

Then from the Laplacian quadratic form, we can now say that two graphs are σ-spectrally similar if

$$L_{\tilde{G}}/\sigma \preceq L_G \preceq \sigma \cdot L_{\tilde{G}}.$$

This tells that finding an approximation of a graph G is equivalent to approximating its Laplacian matrix.

2.3 Sampling via Effective Resistance

We can now further define the goal of spectral sparsification of a weighted graph $G(V, E, w)$ to be finding a sparse subgraph $H(V, \tilde{E}, w')$ where $\tilde{E} \subseteq E$ and H satisfies

$$L_G/(1+\epsilon) \preceq L_H \preceq (1+\epsilon)L_G, \tag{2}$$

with $0 < \epsilon < 1$. Combining this with (1), we can more specifically say that, we want to find a sparse vector s_e where $s_e(e) = w'(e)$ in H and $s_e \geq 0$ that satisfies

$$L_G/(1+\epsilon) \preceq L_H = \sum_{e \in E} s_e b_e b_e^T \preceq (1+\epsilon) \cdot L_G.$$

The way we find this sparse approximation H is by sampling the edges in G. We assign a probability to each edge and sample according to these probabilities. Each edge's probability is determined according to its **effective resistance**, denoted as $Reff(e)$. Intuitively, the effective resistance can be viewed as a metric to denote *how important* each edge is in the graph, or rather, how likely the edge will appear in a random spanning tree of the graph. A high effective resistance indicates that there are few alternative paths to travel between the two vertices of the edge, therefore it is important to include such an edge in the sparse approximation in order to preserve the graph's structure. There are two methods to compute the $effective\ resistance$ of a graph edge.

2.3.1 Graph as an Electrical Network

It turns out that the $effective\ resistance$ of a graph edge has a natural physical meaning. If we view the graph as an electrical circuit where each edge is considered an electric conductance with resistance $1/w_e$. Then 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 [5]. Recall that for an electrical circuit, the total energy dissipation of the network is the the sum of all the squares of the potential differences. Therefore, if we denote a vector x over the nodes in the network to be the potentials at each node, then the energy dissipation of the network ε_G can be written as

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

Note that this equation for energy dissipation is equal to the Laplacian quadratic form of the weighted graph G. This tells us that approximating the graph in the perspective of a physical network is equivalent to the spectral similarity defined in Section 2.2. Since we are using a unit current $I = 1A$, then the effective resistance of an edge is just equal to the energy dissipation of that one edge when the current is injected. In addition, the electrical flow of the network always has the tendency to minimize the overall energy dissipation. Thus, if we inject a unit current on this bridge edge in Figure 2(a), we get the edge's effective resistance to be equal to 1 because it is the only edge the current can flow through between that edge's two endpoints. However, if we inject the current between the vertices in Figure 2(b) and look at the effective resistance of that edge, we will find that its effective resistance will be less than one since some current will flow through this alternative path to minimize the total energy dissipation. Therefore, an edge with a high effective resistance indicates that the edge is electrically important, therefore should be sampled with a higher probability in the approximation. In fact, we will use the the value of an edge's effective resistance to be its sampling probability.

To compute the effective resistances of the edges in the graph, we look at graph G with arbitrarily oriented edges. For a vector $\mathbf{i}_{ext}(u)$ of currents injected at the vertices, let $\mathbf{i}(e)$ be the currents induced in the edges (in the direction of orientation) and $\mathbf{v}(u)$ the potentials induced at the vertices. By Kirchoff's current law, the sum of the currents entering a vertex is equal to the amount injected at the vertex:

$$B^T\mathbf{i} = \mathbf{i}_{ext}.$$

By Ohm's law, the current flow in an edge is equal to the potential difference across its ends times its conductance. Since we previously defined an edge's conductance to be simply its weight w_e, we now have:

$$\mathbf{i} = WB\mathbf{v}.$$

Combining these two facts, we obtain

$$\mathbf{i}_{ext} = B^T(WB\mathbf{v}) = L_G\mathbf{v}.$$

If the total amount of current injected is equal to the total amount extracted – then we can write

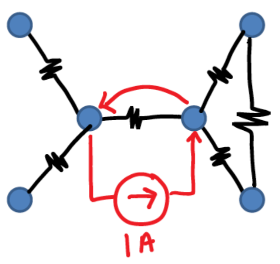
$$\mathbf{v} = L_G^+\mathbf{i}_{ext}$$

by the definition of L^+ in Section 2.1.

Recall that the effective resistance between two vertices u and v is defined as the potential difference induced between them when a unit current is injected at one and extracted at the other. We will derive an algebraic expression for the effective resistance in terms of L_G^+. To inject and extract a unit current across the endpoints of an edge $e = (u, v)$, we set $\mathbf{i}_{ext} = b_e^T = (\chi_v - \chi_u)$. Then the potentials induced by $\mathbf{i}_{ext}$ at the vertices are given by $\mathbf{v} = L^+b_e^T$; to measure the potential difference across $e = (u, v)$, we simply multiply by b_e on the left:

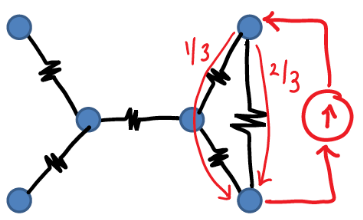
$$\mathbf{v}(v) - \mathbf{v}(u) = (\chi_v - \chi_u)^T\mathbf{v} = b_eL_G^+b_e^T$$

It follows that the effective resistance across e is given by $b_eL_G^+b_e^T$ and that the matrix $BL_G^+B^T$ has as its diagonal entries $BL_G^+B^T(e, e) = Reff(e)$.



Reff$(e) = 1^2 = 1$

(a)



Reff$(e) = (2/3)^2 + (1/3)^2 + (1/3)^2 = 2/3$

(b)

Figure 2: The potential differences induced when injecting and extracting a unit current at the two different edges. We can see the different $Reff(e)$ for the two edges of the network.

2.3.2 Quadratic Forms as Ellipsoids

The second way to find each edge's effective resistance is to look at the quadratic form of G as an ellipsoid. If we only consider the Laplacian matrix on $\text{im}(L_G)$, then we can define a graph's n-dimensional bounded ellipsoid by its quadratic form $x^T L_G x \leq 1$, and all the edges' incidence vectors b_e in $L_G = \sum_{e \in G} b_e b_e^T$ will constitute the level set $Q_G(x) = 1$ of this ellipsoid. In addition, equation (2) can be expressed by this containment relationship of the ellipsoids shown in Figure 3.

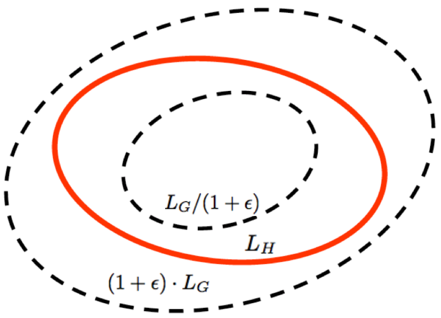


Figure 3: Inequality (2) of the Laplacians expressed as containment of the ellipsoids.

Furthermore, this containment relationship holds under a rescaling of the Laplacian matrices. Namely, the inequality

$$ML_GM/(1+\epsilon) \preceq ML_HM \preceq (1+\epsilon) \cdot ML_GM$$

holds with a symmetric matrix M. Now if we choose $M = L_G^{+1/2}$, then we have

$$\begin{aligned} & L_G^{+1/2} L_G L_G^{+1/2}/(1+\epsilon) \preceq L_G^{+1/2} L_H L_G^{+1/2} \preceq (1+\epsilon) \cdot L_G^{+1/2} L_G L_G^{+1/2} \\ \Rightarrow \quad & I/(1+\epsilon) \preceq L_G^{+1/2} L_H L_G^{+1/2} \preceq (1+\epsilon) \cdot I \end{aligned} \tag{3}$$

because $L_G^{+} L_G$ is equal to the identity matrix on $\text{im}(L_G)$. Since the matrix $L_G^{+1/2} L_H L_G^{+1/2}$ is equal to $\sum_{e \in E} s_e L_G^{+1/2} b_e b_e^T L_G^{+1/2}$, we can rewrite it as $\sum_{e \in E} s_e v_e v_e^T$ where $v_e :=$

$L_G^{+1/2}b_e$ are the rescaled incidence vectors when we rescale the ellipsoid to approximate the identity matrix, whose quadratic form will just turn out to be a sphere. The reason why this procedure will help us find the effective resistances is because when we reshape the ellipsoid to a sphere, all test directions are equally important in multiplicative approximation. Therefore, rescaling will reveal the *important* vectors since their norms will be magnified [11]. Therefore, under the rescaling, we can simply use the norm of the rescaled incidence vectors $\|v_e\|^2 = \|L_G^{+1/2}b_e\|^2 = b_e L_G^+ b_e^T$ to denote an edge's effective resistance. Note that this result is consistent with the one specified in Section 2.3.1.

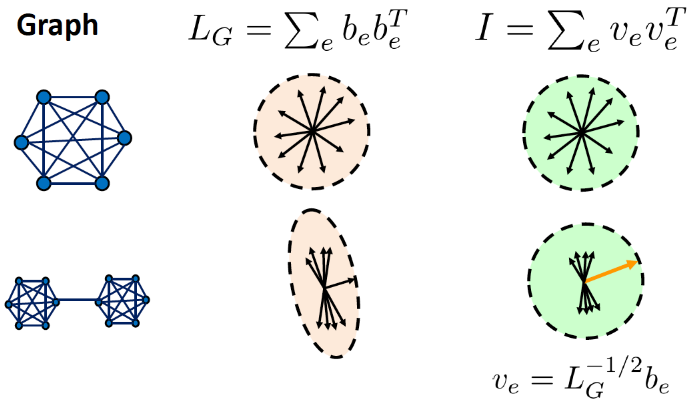


Figure 4: The plotted ellipsoids of a complete graph and a barbell graph. In a complete graph, we know that the *effective resistances* are uniformly distributed across the edges. Therefore when its ellipsoid is recaled to a sphere, its incidence vectors remain the same and their norms are also uniform across edges. On the other hand, since a barbell graph has a bridge edge, its ellipsoid will have a more squashed shape. After rescaling its ellipsoid, however, the norm of the bridge edge's incidence vector will be magnified relative to other edges, revealing that it is an *important* edge [11].

2.4 The Algorithm

Now that we have established ways to compute the effective resistance of each edge, we can formalize Spielman and Strivastava's algorithm of spectral sparsification as follows:

Theorem 1. *To find a ϵ-approximation H of a weighted, undirected graph G, we randomly sample with replacements k edges of G with probabilities $p_e \propto \|L_G^{+1/2}b_e\|^2$ and add the edges to H with new weights $w'_e = 1/kp_e$. Sum the weights if an edge is chosen more than once.*

From the Chernoff Bound, we can further specify that k is at most $9n\log n/\epsilon^2$, and with probability at least $1/2$, the resulting graph H satisfies (2).

2.4.1 Computing Approximate Resistances Quickly

Spielman and Srivastava also introduced an improvement to the algorithm that computes approximations of effective resistances quickly by using the Johnson-

Lindenstrauss Lemma. Recall that an edge's effective resistance

$$\begin{aligned}
Reff_G(e) &= \|L_G^{+1/2} b_e\|^2 \\
&= b_e^T L_G^+ b_e \\
&= b_e^T L_G^+ L_G L_G^+ b_e \\
&= (b_e^T L_G^+ B^T W^{1/2})(W^{1/2} B L_G^+ b_e) \\
&= \|W^{1/2} B L_G^+ (\chi_u - \chi_v)\|^2.
\end{aligned}$$

This suggests that computing the effective resistances of each edge is equivalent to computing the pair-wise distances of the column vectors of the matrix $W^{1/2}BL_G^+$. This calculation's runtime is dependent on the dimension of the column vectors, which in this case is the number of edges, m. The Johnson-Lindenstrauss Lemma states that these Euclidean distances between vectors are approximately preserved if we project the vectors onto a subspace spanned by $O(\log n)$ random vectors. In other words, this dimension-reduction function allows us to bring down the runtime of computing effective resistances from $O(m)$ to $O(\log n)$ [7].

Lemma 1 (Johnson-Lindenstrauss)**.** *For any set S of n-points in d-dimensions, there is a matrix $A \in \mathbb{R}^{k \times d}$ whose entries are independent Gaussian entries such that*

$$\forall u, v \in S, \quad (1-\epsilon)\|u-v\|^2 \le \|Au - Av\|^2 \le (1+\epsilon)\|u-v\|^2,$$

with $k = O(\frac{\log n}{\epsilon^2})$.

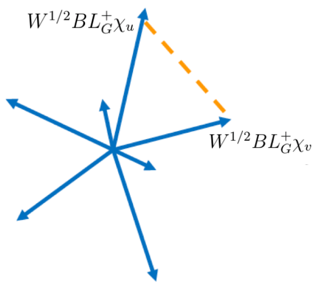


Figure 5: *Effective resistances* are just the pair-wise distances of the column vectors.

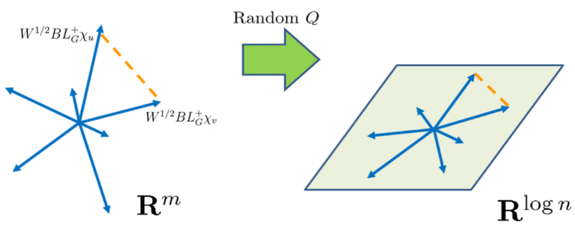


Figure 6: The Johnson-Lindenstrauss Lemma states that these distances will be preserved approximately when the matrix is mapped onto $\log n$ dimension by the transformation matrix Q whose entries are of Gaussian distribution.

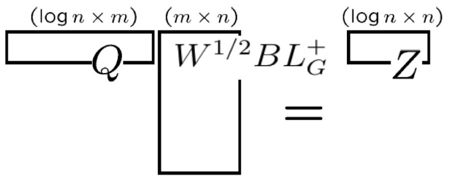


Figure 7: We can use the matrix $Z = QW^{1/2}BL_G^+$ to calculate approximations of the *effective resistances* quickly [11].

Therefore, the refined algorithm of spectral sparsification is as follows:

Algorithm 1: Spectral Sparsification

```
    Input: weighted, undirected graph $G = (V, E, w)$ with $n$ vertices and $m$ edges
    Output: sparse subgraph $H = (V, \tilde{E}, w')$
1   Compute a random matrix $Q_{logn \times m}$ where the entries of $Q$ are of Gaussian
      distribution.
2   Compute $Z = QW^{1/2}BL_G^+$.
3   Set a value of $k$ such that $k \leq 9n\log n/\epsilon^2$.
4   for $i = 1$ to $k$ do
5       Assign $p_e \propto \|Z(\chi_u - \chi_v)\|^2$ to each edge in $G$;
6       Sample $e \in G$ with replacements with probability $p_e$;
7       if $e$ is selected then
8           add $e$ to $H$ with new weight $w'_e = 1/kp_e$;
9           sum up weights if $e$ is selected more than once
```

3 Implementation and Evaluations

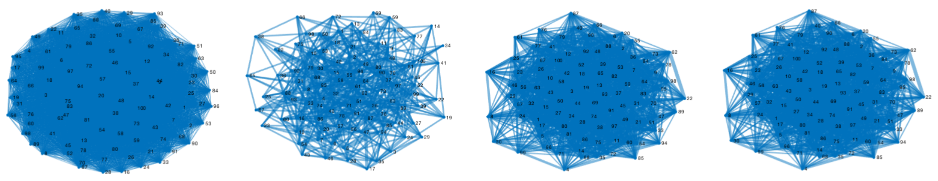
In this section, I will present some empirical results from my implementation of the sparsification scheme. Upon reviewing their theoretical framework, I implemented Spielman and Srivastava's spectral sparsification using MATLAB scripts and tested the algorithm on both generated and real-world graphs. I was mainly curious about the effect of the number of edges sampled on the accuracies of the approximations. All test graphs are stored in .gml files and then parsed in MATLAB. Since MATLAB is very well-equipped with tools for graph and matrix computations, calculating the incidence and Laplacian (including its *Pseudoinverse*) matrices was a straightforward procedure. Furthermore, I only employed the simple implementation to calculate effective resistances since the computation of the matrix BL_G^+B is rather trivial in MATLAB. Therefore, my experiment results are not conclusive on the accuracy of the overall sparsification using approximated effective resistances.

Specifically, my synthetically-generated test graphs include a complete graph and a barbell graph, both of 100 vertices. This was done using the NetworkX package in python. I chose a complete graph simply because it is the most typical and simple dense graph that provides a good platform for sparsification and easy ways to analyze. As shown in Figure 8, since a complete graph is the densest graph out there, there is quite a lot of room to sparsify it. Therefore, my k value was actually able to vary from $1 \cdot n\log n$ to $7n\log n$. Figure 8 shows the original complete graph and 3 sparsified approximations with $k = n\log n$, $2n\log n$, and $3n\log n$. From the figure, one can tell that all 3 approximations preserve the general structure of the original graph decently well without losing connectivities.

Subsequently, I generated a barbell graph (two identical complete graphs connected by a single edge in between) as my second test case. I chose a barbell graph because it also has special structural properties that could help us test the correctness of the algorithm. Since in a barbell graph, the edge connecting the two complete graphs is the bridge edge whose removal will disconnect the graph, it should have an effective resistance of 1 and always appear in the sparsification after random sampling. My implementation results confirmed this expectation of Spielman and Srivastava's theory. As shown in Figure 9, the two complete graphs at the two ends of the graph behave similarly as the first case during sparsification and the connecting bridge edge consistently appears in the approximation with its new weight equal to 1.

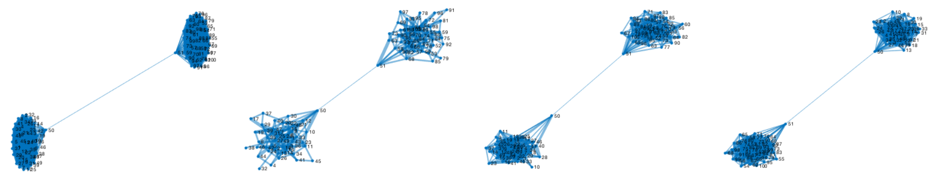
In order to examine the practicality of the algorithm in real-world applications, I additionally ran the implementation on two graphs drawn from real world data. The first graph is a network data of American football games between Division IA colleges during regular season Fall 2000 with $n = 115$ and $m = 613$ [6]. The second graph is the adjacency network of common adjectives and nouns in the novel *David Copperfield* by Charles Dickens with $n = 112$ and $m = 425$ [9]. Again, both graphs are stored in .gml format. Sparsifying these real world graphs proves to be a bit more challenging than the two sythetic graphs. This is inevitably due to the fact that most real world natural graphs are already pretty sparse! Therefore, further sparsifying these graphs will very easily result in losing some degrees of connectivies. This effect is shown in both Figure 10 and Figure 11 and is consistent across the two natural graphs. Note that Spielman and Srivastava's theorem states that we are able to sparsify any graphs [12]. However, the value of k, i.e., the number of edge samples to include in the approximation needs to be carefully chosen. If we include too few edges, the resulting approximation tends to lose a significant amount of connectivities, which is certainly not desired since it changes the fundamental structure of the graph. This also generally means a less accurate aprroximation, i.e., a high ϵ value. However, if we choose the value of k to be too close to m, we may be sacrificing the degree of sparseness which defeats our original purpose.

Finally, in all experiment cases, I have found that the algorithm provides a reasonable sparsification with $\epsilon < 1$, which is an anticipated computational behavior. In addition, the resulting values of ϵ decreases consistently as k increases (Figure 12). This indicates that sampling more edges provides a more accurate approximation. This is also an expected result since theoretically, as k gets closer to m, the output graph becomes closer to the original.



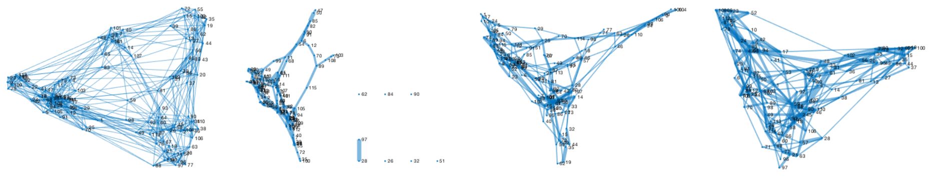
original $k = n\log n$ $k = 2n\log n$ $k = 3n\log n$

Figure 8: Sparsification results for a complete graph with 100 vertices.



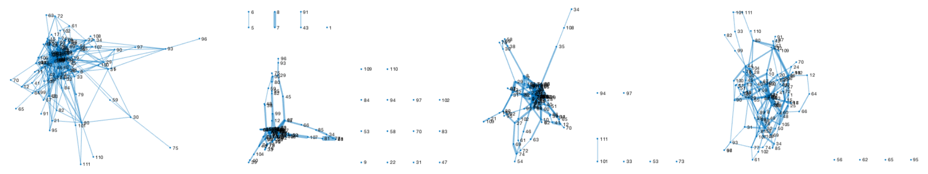
original $k = 0.5n\log n$ $k = n\log n$ $k = 1.5n\log n$

Figure 9: Sparsification results for a barbell graph with 100 vertices.



original $k = 0.2n\log n$ $k = 0.4n\log n$ $k = 0.6n\log n$

Figure 10: Sparsification results for football.gml.



original $k = 0.2n\log n$ $k = 0.3n\log n$ $k = 0.4n\log n$

Figure 11: Sparsification results for adjnoun.gml

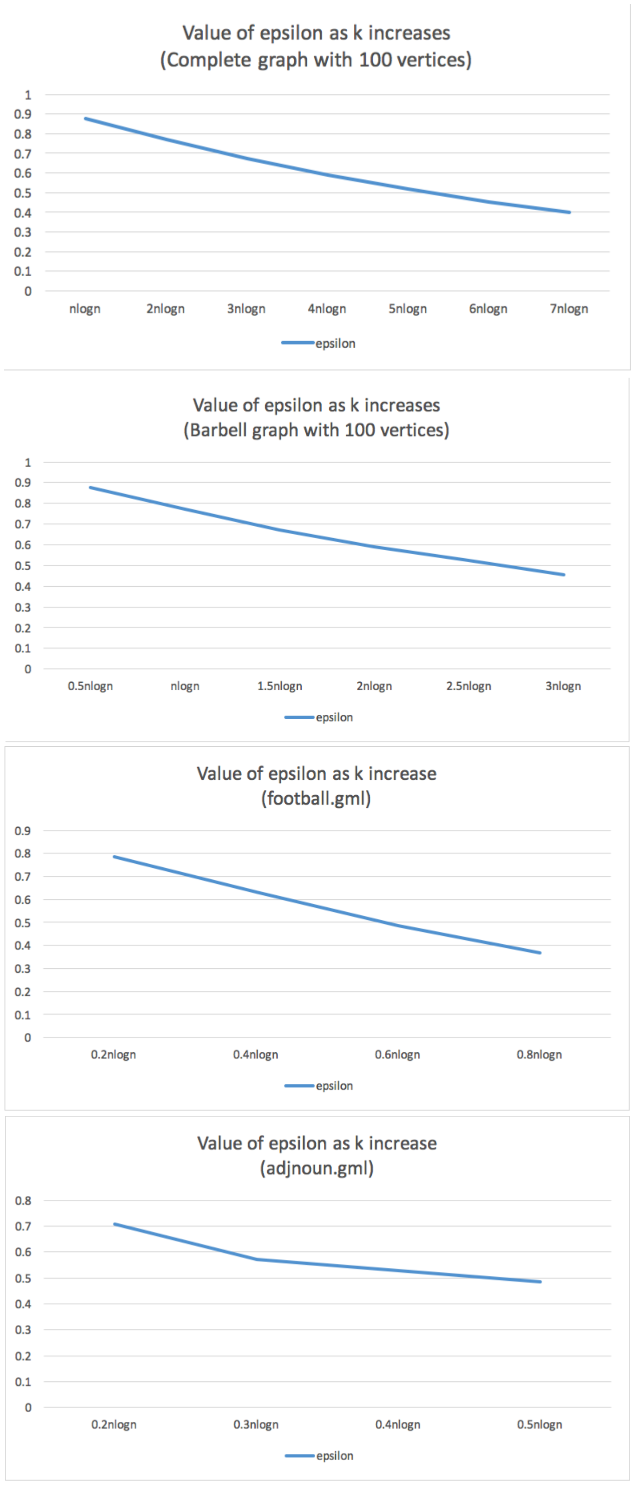


Figure 12: Resulting values of ϵ as the value of k changes.

4 Extension on Multigraph Models

We have illustrated that Spielman and Srivastava's sparsification algorithm applies neatly to simple, undirected graphs. However, certain problems are more easily analyzed if modeled with complex graphs as multigraphs or hypergraphs. A multigraph is defined as a typeof graph where there are more than one edges connecting the same pair of vertices anda hypergraph is one where an edge can connect more than two vertices. It could be veryuseful if we can apply Spielman and Srivastava's sparsification algorithm to multigraphsand hypergraphs with reasonable modifications. In this section we give a proof that the $effective\ resistance$ of a parallel edge in a multigraph can be calculated using the same method as in a simple graph.

Theorem 2. *Let $e_1, e_2, e_3, \ldots, e_k$ be the parallel edges with identical weights w between vertices a and b in Graph G and $\mathbf{R}(a, b)$ be the effective resistance between a and b. Then the effective resistance on each edge $\mathbf{R}(e_j) = \frac{1}{k}\mathbf{R}(a, b)$.*

Proof. Let $\mathbf{i}_j$, $j = 1, 2, \ldots, k$ be the electric current that flows through one of the parallel edges e_j between nodes a, b and $\mathbf{i}_{ext}$ be the total amount of current injected at vertex a. From Kirchoff's current law, we have

$$\mathbf{i}_{ext} = \sum \mathbf{i}_j.$$

Since edges $e_1, e_2, \ldots, e_k$ are of identical weights w, they have the same conductance,

$$\begin{aligned}
\Rightarrow \qquad & \mathbf{i}_1 = \mathbf{i}_2 = \ldots = \mathbf{i}_k \\
\Rightarrow \qquad & \mathbf{i}_j = \frac{1}{k}\mathbf{i}_{ext}, \qquad j = 1, 2, \ldots, k.
\end{aligned}$$

From the original proof in Spielman and Srivastava's paper, we have

$$\mathbf{v} = L^{+}\mathbf{i}_{ext},$$

where $\mathbf{v}$ is the voltage induced at node a, and

$$\mathbf{R}(a, b) = b_e L^{+} b_e^T.$$

If we denote the potential at one of the endpoints of edge e_j that is responsible for only current $\mathbf{i}_j$ as $\mathbf{v}_j$, then we have

$$\begin{aligned}
\mathbf{v}_j &= L^{+}\mathbf{i}_j \\
\Rightarrow \qquad &= L^{+}(\frac{1}{k}\mathbf{i}_{ext}) \\
\Rightarrow \qquad &= \frac{1}{k}\mathbf{v}.
\end{aligned}$$

Finally, we simply multiply by b_e on the left to obtain the potential difference:

$$\begin{aligned}
\mathbf{R}(e_j) = \mathbf{v}_j(a) - \mathbf{v}_j(b) &= b_e\frac{1}{k}\mathbf{v} \\
\Rightarrow \qquad &= \frac{1}{k}b_e L^{+} b_e^T \\
\Rightarrow \qquad &= \frac{1}{k}\mathbf{R}(a, b).
\end{aligned}$$

□

5 Related Work

It might also be worth mentioning some other approaches to graph sparsification. One interesting altenative to Spielman and Srivastava's spectral sparsification using 'importance' sampling is the sparsifying by approximations of all-pairs distances of the graph [2]. This type of sparsified approximation is called a *t-spanner*, which is defined to be a subgraph $\tilde{G}$ of G on the same vertex set that satisfies

$$dist_{\tilde{G}}(u,v) \leq t \cdot dist_G(u,v)$$

for all $u,v \in V$ where $dist_G(u,v)$ is the distance between vertices u and v [10]. The parameter t is called a $stretch\ factor$. This approach of sparsification comes down to finding good sparse *t-spanners*. Althofer et al. have shown that that every weighted graph has a $(2t+1)$-*spanner* with $O(n^1 + 1/5)$ 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, up to $O(m\log n\log\log n)$ distortion [1].

6 Conclusion and Future Work

By examining the empirical results from my implementation of Spielman and Srivstava's sparsification scheme, I conclude that their algorithm produces anticipated results for selected graphs. Furthermore, I believe this approach will be a significantly useful tool for efficient graph computations. We should also be able to apply the graph sparsification scheme to develop faster algorithms of solving systems of linear equations.

However, there are definitely ways to improve the algorithm. One major issue with the current algorithm is the concern about the sparseness of most real-world graphs. As previously demonstrated, when the input graph is reasonably sparse to begin with, the value of k needs to be carefully determined to produce a decent approximation while not distorting the properties of the original graph. Note that even for a sparse graph input, we would still like to sparsify it if possible since most graph algorithms such as traversals greatly depend on the number of total edges. From our experiments, there is clearly a trade-off between the sparseness and the accuracy of the approximation. Therefore, the question naturally becomes: what would be a reasonable range of k? Or in other words, for relatively sparse graphs with edges significantly fewer than $9n\log n$, could we establish a lower bound of the number of edges to sample to guarantee a particular upper bound of ϵ and a certain degree of sparseness? In addition, we would like to examine the impact of a decent graph sparsification on problems such as graph partitioning and clustering. I hope that this body of work lays the ground for further development of faster graph algorithms.

References

[1] Ittai Abraham and Ofer Neiman. Using petal-decompositions to build a low stretch spanning tree. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 395–406, New York, NY, USA, 2012. ACM.

[2] Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, Jan 1993.

[3] Thomas N.E. Ben-Israel, Adi; Greville. *Generalized inverses: Theory and applications.* New York, NY: Springer, 2003.

[4] András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $\tilde{O}$(n2) time. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 47–55, New York, NY, USA, 1996. ACM.

[5] Ashok K. Chandra, Prabhakar Raghavan, Walter L. Ruzzo, Roman Smolensky, and Prasoon Tiwari. The electrical resistance of a graph captures its commute and cover times. *computational complexity*, 6(4):312–340, Dec 1996.

[6] M. Girvan and M. E. J. Newman. Proc. natl. acad. sci. *USA 99*, 2002.

[7] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26:189–206, 1984.

[8] J. McAuley and J. Leskovec. Learning to discover social circles in ego networks. *NIPS*, 2012.

[9] M. E. J. Newman. Phys. rev. *E74*, 2006.

[10] David Peleg and Jeffrey D. Ullman. An optimal synchronizer for the hypercube. *SIAM Journal on Computing*, 18(4):740–747, 1989.

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

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