Evaluation of Sparsification by Effective Resistances

Benjamin Hollander-Bodie Sam Leone David Metrick Crystal Wang

Link: $https : //www.youtube.com/watch?v = K_o0Kw9j5xw$

***Abstract*—In this paper, we implement and evaluate the algorithm proposed in Sparsification by Effective Resistances. We show that for sufficiently large samples of edges, the eigenvalues of sparsified graphs do approach those of the original graph, and their corresponding Laplacian quadratic forms are also close. We also demonstrate this sparsification is superior to K-nearest-neighbors as a clustering scheme for denoising the edges of a graph. We also show how at sufficiently large scales of data, the sparsification preserves geometric information, such as the set of diffusion curvatures at each vertex. Finally, we provide an illustration of why the sparsification is not well suited for TDA.**

I. Introduction

In modern data science, graphs have been employed as a tool for representing data of great generality. For example, an $N \times M$ image can be viewed as existing on a $N \times M$ grid graph, with edges between adjacent pixels. In a social platform, we could have a graph of individuals, with edges between friends. Graph signal processing has also had success in considering a graph of traffic data, the human brain, and point cloud data. Likewise, the field of manifold learning treats data sets as point clouds in high dimensional space, and induces a graph by calculating pairwise affinities, often by running the set of pairwise Euclidean distances through a nonlinear & symmetric kernel.

Furthermore, from graphs come far-reaching applications. For example, the min-cut of a graph provides a measure of the severity of bottlenecking in the data. A related notion of graph conductance is an adjustment of min-cut in a way which is sensitive to the sizes of the partition of the vertices. Graph based algorithms like Louvain clustering [2] provide modes of neighborhood detection in the data. Related to all of these is the eigendecompositon of the so-called Graph Laplacian, to be defined. Spectral Graph Theory as a field of study relates the eigenvalues and eigenvectors of this matrix to the geometric and algebraic properties of the graph [7]. The second eigenvector of this matrix is known as the Fiedler vector, and can be used to partition a graph [3]. The corresponding eigenvalue λ_2 is known as the spectral gap, and is related to the conductance of a graph by the celebrated Cheeger's Inequality (first proved for manifolds by Jeff Cheeger, and subsequently in the discrete setting in the late 80's). Extending the notion of partitions via eigenvector values, the spectral clustering algorithm works by using the first k eigenvectors of the Laplacian and performing k-means in this embedded space [6]. Similarly, Kenneth Hall suggests drawing a graph in k-dimensional space in a way which minimizes the Euclidean distance between neighbors in the graph, which can be accomplished by using the first k eigenvalues of the graph as coordiantes in k-dimensional space for each vertex [4]. The premise of Graph Signal Processing is to transform signals on graphs into the basis of graph frequencies and augmenting the weights in some sort of low or high pass filter [5]. This is all to say, the eigenvalues and eigenvectors of a graph has far-reaching applications.

However, many of these algorithms, particularly for large graphs, can be extremely expensive to compute. Clearly, the runtime of a graph-based algorithm will depend both on the number of vertices and the number of edges. Eigendecomposition of a dense graph occurs in time roughly $\mathcal{O}(n^3)$ without an assumption on graph sparsity. A graph with n vertices could have as many as $n(n-1)/2$ edges. And if it has fewer than $n-1$ edges, the graph is disconnected, and can for all purposes be regarded as a set of graphs. Thus, the number of edges in a graph will always dominate the number of vertices. And so sparse graphs (with $\mathcal{O}(n \log n)$ edges) will maximize efficiency. For a given dense (not sparse) graph, we are therefore interested in approximating our graph well using as few edges as possible, in order to accelerate processes down the road. This is the process of graph sparsification.

There are many ways to sparsify a graph. One such process is known as k-nearest-neighbors, in which we only keep sufficiently large affinities from our original graph. This is an intuitive move, but doesn't generalize well outside of the geometric setting. Furthermore, for general graphs, how can one even assess how well a sparsification approximates the original? The idea of Spielman & Srivastava [8] is to use the graph Laplacian to capture similarity, such that sparsified graphs have a similar spectrum.

II. Background

A. The Graph Laplacian

A weighted graph $G = (V, E, w)$ is a triple of vertices, edges, and weights $w : E \to \mathbb{R}^+$; here we assume weights represent nonnegative affinties between vertices. Let the vertices of G be $V = \{1...n\}$. The adjacency matrix A of G has $A(a,b) = w(a,b)$. The degree of vertex a is $\deg(a) \triangleq \sum_{(a,b)\in E} w(a,b)$. The degree matrix of G is the diagonal matrix D with $D(a,a) = \deg(a)$. The combinatorial Graph Laplacian of G is $L \triangleq D - A$. It is a well known fact about the Laplacian that, for a vector $x \in \mathbb{R}^n$

$$x^T L x = \sum_{(a,b)\in E} w(a,b)(x(a)-x(b))^2 \qquad (1)$$

Furthermore, L is symmetric, and so by the spectral theorem, it is diagonalizable with an orthonormal basis of eigenvectors. By Equation 1, L is positive-semidefinite, and so its eigenvalues are all nonnegative. One can also easily verify that if ψ is an eigenvector of L of eigenvalue λ, then $\psi^T L\psi = \lambda$. And so, the eigenvalues correspond to total variation of the corresponding eigenvector over the graph; so eigenvectors of low eigenvalues are said to be smooth. Furthermore, we find that the quadratic form $x^T Lx$ is intimately connected with the graph spectrum, and so we define a measure of graph similarity based on this. We say, for two $n \times n$ matrices A and B, $A \preceq B$ if $x^T Ax \leq x^T Bx$ for all $x \in \mathbb{R}^n$. Likewise, if L_G is the Laplacian of G and L_H is the Laplacian of H, we say $G \preceq H$ if $L_G \preceq L_H$.

B. *Effective Resistances*

Consider an interpretation of G as a system of electrical flows, in which the adjacency $w(a,b)$ is the conductance of an edge. The *effective resistance* of two vertices a and b is defined to be the potential difference between them when one unit current is injected at a and extracted at b, or the voltage between a and b. Let i_{ext} be the vector of voltages injected at the vertices, i is the corresponding current at each edge, and v is the vector of voltages. Then by Kirchhoff's current law, we know that,

$$\sum_{(a,b)\in E} i(a,b) = i_{ext}(a)$$

Where the left hand side is the sum of voltages leaving a and i_{ext} is the amount entering a. If the (unweighted) incidence matrix of G is B, where, for an arbitrary orientation of the edges, $B(e,v)$ is 1 if v is the head of e and -1 if v is the tail (and 0 otherwise). This condition becomes equivalent to $B^T i = i_{ext}$. But also, Ohm's Law states that the flow at each edge is equal to the potential difference times conductance. As voltages are the vector of potential differences, this becomes,

$$i = WBv$$

Where W is the diagonal matrix such that $W(e,e) = w(e)$. And thus, we have,

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

Where $B^T WB = L$ by a simple algebraic verification. And thus, the vector of voltages is provided by $v = L^\dagger i_{ext}$ (if i_{ext} has mean 0), where $L^\dagger$ is the Moore-Penrose pseudoinverse of L. Thus, to determine the effective resitance of edges a and b, we let $i_{ext} = \delta(a) - \delta(b)$, so that we inject 1 unit of current at a and remove it at b. And then, the effective resistance of a and b is provided as

$$v(a) - v(b) = \delta(a)v - \delta(b)v = i_{ext}^T v = i_{ext}^T L^\dagger i_{ext} \qquad (2)$$

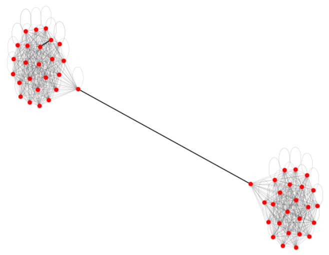


Fig. 1. Effective Resistance of each edge on the dumbbell graph with $n = 25$. An electrical flow between the components would involve heavy current through the bar.

Thus, we define the effective resistance between vertices a and b to be $Re(a,b) = (\delta(a) - \delta(b))^T L^\dagger(\delta(a) - \delta(b))$. Likewise, we define the effective resistance of an edge e with endpoints a, b to be $Re(e) = Re(a,b)$.

III. Methods

A. *Implementing the Spielman-Srivastava Algorithm*

Algorithm 1 Algorithm For Sampling Effective Resistances

```
  $H$ is an empty graph with the same vertices as $G$
  for $k \in \{1...q\}$ do
      Choose a random edge of $G$ with probability $p(e)$ proportional to $w(e)R(e)$
      Add $e$ to $H$ with wight $w(e)/qp(e)$, summing weights if an edge is chosen more than once
  end for
  return $H$
```

Spielman & Srivasta [8] present a randomized algorithm for subsetting edges, based on the effective resistance of those edges. They present the following algorithm for sampling effective resistances. What Spielman and Srivastava show is that, through a suitable choice of parameters, given an $\epsilon > 0$, this algorithm will produce with high probability a graph H with $\mathcal{O}(n \log n/\epsilon^2)$ vertices such that, $(1-\epsilon)G \preceq H \preceq (1+\epsilon)G$, i.e. H is said to be an ϵ-approximation of G. Note the exchange between quality of approximation and runtime. Observe that, in order to execute this algorithm, we need to know the effective resistances of the edges. But observe that equation 2 is suggestive of calculating the pseudoinverse $L^\dagger$, which requires diagonalization of L, which is a $\mathcal{O}(n^3)$ task. The authors suggest an approximation algorithm of the effective resistances, noting the resistances can be reviewed as a Euclidean distance in a suitable space, making an appeal to Johnson Lindenstrauss to reduce the dimensionality of that space, and then rapidly solving a system of linear equations. In particular, we observe, letting $b_e = \delta(a) - \delta(b)$ for an edge $e = (a,b)$

$$b_e^T L^\dagger b_e = b_e L^\dagger L L^\dagger b_e = b_e^T L^\dagger B^T W^{1/2} W^{1/2} B L^\dagger b_e$$
$$= \|W^{1/2} B L^\dagger b_e\|^2$$

Due to a version of the Johnson-Lindenstrauss Theorem of Achlioptas, these distances can be approximately preserved by multiplication of $W^{1/2}BL^{\dagger}b_e$ with a random matrix $Q \in \mathbb{R}^{k\times n}$ (for a suitable $k < n$), in which $Q(i,k)$ is random Bernoulli either $1/\sqrt{k}$ or $1/\sqrt{k}$. Then, letting $Z = QW^{1/2}L^{\dagger}$ computing $QW^{1/2}L^{\dagger}$ corresponds to solving the k equations $ZL = QW^{1/2}$ (where view each row of Z as an unknown, and solve the linear system corresponding to L). Because of the dimensionality reduction, this requires solving only k, rather than n, linear equations. Spielman & Teng [9] also present an asymptotically near-linear algorithm for solving these equations. But for the scope of this project, we appeal to ordinary sparse solvers, such as Cholesky decomposition or the conjugate gradient method. Thus, we run the following algorithm to estimate the resistances.

Algorithm 2 Estimating Effective Resistances

```
Parse $L$ to find $W^{1/2}$ and $B$
Randomly generate $Q \in \mathbb{R}^{k\times n}$ as specified
Set $Y = QW^{1/2}B$
Solve $ZL = Y$
For each $e = (a,b)$ set $R(e) = \|Z\delta(a) - Z\delta(b)\|^2$
```

Note this algorithm has user input ϵ and hyperparameters k and q. k is prescribed as $24\log n/\epsilon^2$, and is clearly much less than n for fixed ϵ and sufficiently large graphs. However, q is provided only asymptotically, and so we turn it into a user parameter (in doing so, we eliminate the theoretical guarantees of the algorithm on small problems, while providing greater user flexibility). Alternatively, to calculate the effective resistances, we can also estimate $b_e^T L^{\dagger} b_e = b_e^T L^{\dagger/2} L^{\dagger/2} b_e = \|L^{\dagger/2} b_e\|^2$ through Chebyshev polynomials of the graph Laplacian, corresponding to a filter of $h(\lambda) = \lambda^{-1/2}$. We also provide the optionality for this approximation, which avoids the dimensionality reduction. Comparison of these methods on toy datasets reveals their consistency.

IV. Experiments & Results

A. Performance on the Dumbbell Graph

A sanity check for the importance of resistances is the so-called dumbbell graph. This graph G on n vertices consists of two separate copies of the complete graph on n vertices K_n, with just one edge connecting the two. In this case, it is obvious that any electric flow between vertices in opposite ends of the graph will result in a high level of current flowing through the connecting edge, with relatively little happening elsewhere. And thus, sparsification of the dumbbell graph will involve removing edges almost arbitrarily from the fully connected subgraphs (by symmetry), while preserving the main edge, upon which the connectivity hinges.

In this experiment, we generate a graph sparsification of the dumbbell graph with $n = 25$ via Algorithm III-A. We use $k = 50$ as the projected dimension of the resistive embeddings. Then, for each of $q = 150, 250, 300, 1000, 20000, 100000$, we generate a sparsifid graph. Naturally, we would hope that by the law of large numbers, as $q \to \infty$, the quality of the sparsification grows. Indeed, for each q, we find the correlation between the eigenvalues of original and sparsified graph H_q (here we let H_q denote the subgraph generated by parameter q). We also generate 1000 random unit vectors and find the maximum value of $|x^T L_G x - x^T L_{H_q} x|$, which we call "$\epsilon$"-closeness; the sparsification implies $\sup_{|x|=1} x^T L_G x - x^T L_H x \to 0$. We also compute the effective resistances directly using the pseudoinverse, and find it is 94.8% correlated with the estimated effective resistances. And they are 94.0% correlated with the resistances estimated by Chebyshev polynomials.

	Correlation	ϵ-Closness
150	0.355240	11.643839
200	0.393768	10.767303
300	0.474376	8.371558
1000	0.626117	5.298738
20000	0.972060	0.973045
100000	0.992920	0.453463

Fig. 2. Results of experiment: we find that as q increases, the quality of the approximation increases arbitrarily high. We find that the correlation between eigenvectors nears 1, and the ϵ-closeness tends to 0. Of course, when $q >> m$, sparsification is not truly being accomplished. Rather, this is a sanity check for the sampling procedure.

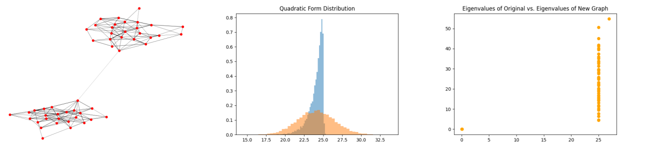


Fig. 3. With $q = 200$, the new sparsified graph is visualized on the left. In the middle, the distribution of $x^T L_G x$ is visualized as a blue histogram, where x are random unit vectors; superimposed is the distribution of $x^T L_H x$. On the right is the scatter plot of eigenvalues of the original against eigenvalues of the new graph.

B. Preservation of Laplacian Eigenmaps

Following the method detailed in [1], we took a base dataset of 1000 points along a Swiss roll and made a weighted affinity matrix W. We first imposed edges onto it using a kth-nearest-neighbor algorithm ($k = 20$), then created an affinity matrix by weighting these edges with the heat kernel

$$e^{||x_i - x_j||^2/\sigma},$$

where σ is the average distance from a point to its kth-nearest neighbor. Afterwards, we reduced the edges of the graph by

1) a KHW (kth-highest weights) algorithm, selecting the edges with the 10th-highest weights for each point in W,
2) running SER (sparsification by effective resistances) on W using default settings, and
3) running SER on W with edges limited to < 4000.

Afterwards, we took the transformed Ws and used them to generate Laplacians and the first two Laplacian eigenvectors

v_1 and v_2. We evaluated these two eigenvectors against those of the original roll using the following metrics:

- L1 norm: the L1 norm of the vector formed by taking the element-wise difference of the original and sparsified eigenvectors, aka $\sum_{i=1}^{1000} |\text{OG\_v1[i]-SPARSE\_v1[i]}|$ and $\sum_{i=1}^{1000} |\text{OG\_v2[i]-SPARSE\_v2[i]}|$
- L2 norm: the L2 norm of the vector formed by taking the element-wise difference of the original and sparsified eigenvectors, aka $\sum_{i=1}^{1000} |\text{OG\_v1[i]-SPARSE\_v1[i]}|^2$ and $\sum_{i=1}^{1000} |\text{OG\_v2[i]-SPARSE\_v2[i]}|^2$
- relative difference: the average percent difference of each element in the original and sparsified eigenvectors, aka $\frac{1}{1000} \sum_{i=1}^{1000} |(\text{OG\_v1[i]-SPARSE\_v1[i]})/\text{OG\_v1[i]}|$ and $\frac{1}{1000} \sum_{i=1}^{1000} |(\text{OG\_v2[i]-SPARSE\_v2[i]})/\text{OG\_v2[i]}|$

Afterwards, we introduced noise into the Swiss roll to evaluate how SER versus KHW dealt with noise.

From a purely visual perspective, SER performed much better than KHW in preserving the immediate structure of the Swiss roll.

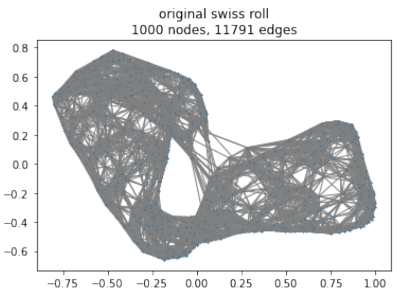
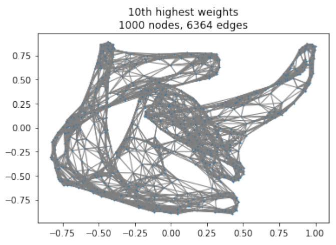
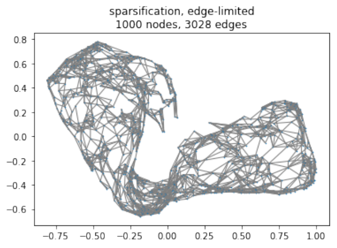


Fig. 4. original Swiss roll with 20th nearest neighbors



(a)



(b)

Fig. 5. graph representation of Swiss roll through (a) KHW (b) SER

However, this similarity did not extend to the Laplacian eigenmaps. Our first finding was that under all metrics, KHW was more effective at preserving Laplacian eigenvectors than SER. However, SER's performance improved under an edge constraint, bringing its performance to a level at least comparable with KHW.

Finally, we found that SER ($|E| < 4000$) did significantly better than KHW in terms of denoising a graph, especially at higher levels of noise. We generated Ws for Swiss rolls at noise levels $\gamma = 0.2, 1$, and 2. Afterwards, we reduced edges using KHW ($k = 10$) and SER ($|E| < 4000$). We then compared the Laplacian eigenvectors for these graphs to the eigenvectors of the original, no-noise Swiss roll.

TABLE I
LAPLACIAN EIGENMAP PRESERVATION, NO NOISE

	reduction method		
metric	*KHW*	*SER*	*SER*($|E| < 4000$)
L1, v_1	1.039	20.75	3.629
L1, v_2	1.310	19.44	3.527
L2, v_1	0.03817	0.7775	0.1335
L2, v_2	0.04752	0.7793	0.1403
rel diff, v_1	12.71%	170.5%	30.73%
rel diff, v_2	27.91%	171.1%	36.71%

These experiments suggest that in terms of Laplacian eigenmaps, SER may perform better than KHW on real, noisier datasets but not on clean or toy data.

TABLE II
LAPLACIAN EIGENMAP PRESERVATION, $\gamma = 0.2$

	reduction method	
metric	*KHW*	*SER*($|E| < 4000$)
L1, v_1	17.37	14.74
L1, v_2	16.12	13.44
L2, v_1	0.6752	0.5792
L2, v_2	0.6293	0.5400
rel diff, v_1	208.4%	156.9%
rel diff, v_2	289.5%	196.4%

TABLE III
LAPLACIAN EIGENMAP PRESERVATION, $\gamma = 1$

	reduction method	
metric	*KHW*	*SER*($|E| < 4000$)
L1, v_1	18.46	15.19
L1, v_2	16.80	13.83
L2, v_1	0.7198	0.5925
L2, v_2	0.6700	0.5542
rel diff, v_1	224.3%	165.4%
rel diff, v_2	300.6%	217.7%

TABLE IV
LAPLACIAN EIGENMAP PRESERVATION, $\gamma = 2$

	reduction method	
metric	*KHW*	*SER*($|E| < 4000$)
L1, v_1	17.40	14.69
L1, v_2	18.56	14.72
L2, v_1	0.6714	0.5629
L2, v_2	0.7050	0.5759
rel diff, v_1	218.6%	165.0%
rel diff, v_2	337.6%	238.4%

C. *Preservation of Topological Properties*

Note that this algorithm focuses on *spectral* sparsification of a graph. In other words, we are truly only approximating the Laplacian quadratic form; but we have no guarantee that this preserves all desirable properties of a graph. Notably, we can see how this sparsification impacts the toplogical properties of a graph, which are the objects of interest in toplogical data analysis (TDA). At a high level, the Betti numbers of a topological structure capture its "holes" in each dimension. A simplical complex is to be thought of as a collection of sets, closed under subsetting. Simplical complexes consist of points, edges, triangles, tetrahedra, etc. in increasing dimension $0, 1, 2, 3$, etc. The pth betti numbers looks at the number of generators of the pth homology group, which can be thought of as the dimension p simplicial cycles which are not the boundary of a $p+1$-dimensional simplex. In this sense, an empty triangle has a loop, because it is a dimension 1 cycle which is not the boundary of existing 2-simplex, and so we say $\beta_1 = 1$. And an empty tetrahedron, a dimension 2 cycle, is not the boundary of an existing 3-simplex, so $\beta_2 = 1$, etc. Thus β_0 counts the number of connected components of a simplcial complex, β_1 the number of loops, and β_2 the number of voids. We can generate simplices from a graph by inducing 1-simplices for edges, 2-simplices for triangles, and 3-simplices for K_4's, and so on. By doing so, we can calculate the corresponding Betti numbers.

In this experiment, we begin with 1000 points sampled on a torus of outer radius 10 and with a loop of radius 4. Then, we generate the simplical complexes with edges between vertices whose distance is less than $\epsilon = 4$, and generate the simplical complexes in the suggested way (this would also coincide with the Vietoris-Rips complex with $\epsilon = 4$). Indeed, the Betti numbers of this graph with 37,224 edges are $\beta_0 = 1, \beta_1 = 2, \beta_2 = 1$, which coincides with our prior knowledge of the torus, which has 1 connected component, 2 nontrivial loops, and just 1 inner void.

Now, with this adjacency matrix (corresponding to $\epsilon = 4$), we run the Spielman & Srivistana sparsification on the graph. Again, we consider the simplicical complex induced by the new set of adjacencies (we binarize the result). If we sparsify the graph with $q = 28,391$, we can calculate the resulting Betti numbers: $\beta_0 = 1, \beta_1 = 2, \beta_2 = 63$. Indeed, we find that we are still succesfully detecting the loops and connected components, although our calculations go horribly wrong on voids. If we sparsify further and set $q = 5000$, the resulting Betti numbers are $\beta_0 = 1, \beta_1 = 1805, \beta_2 = 1$. We conclude this form of sparsification is not well-suited to topological data analysis, which relies very heavily on the geometry of the data; that is, by deleting edges, although we may change the spectrum of the graph by possibly very little, the construction of simplices relies heavily on full connectivity of neighborhoods and thus does not bode well with randomness.

D. Preservation of Diffusion Curvature

In this experiment, we take on the same torus as before, and now examine the diffusion curvature. Recall the diffusion curvature at x is defined to be,

$$C(x) = \sum_{y \in B_r(x)} \frac{m_x(y)}{|B_r(x)|}$$

In this experiment, we take the diffusion curvature at every point using the sparse graph H_q for each $r \in 1, 2, 3, 4, 5$, $q \in 10,000, 20,000, 30,000$. We also take diffusion curvature according to the usual graph at each vertex. And for each r and q, we see how well correlated the resulting total set of diffusion curvatures are between the unsparsified and sparsified graphs. Indeed, we find that as q and r increase, the quality of approximation increases as well.

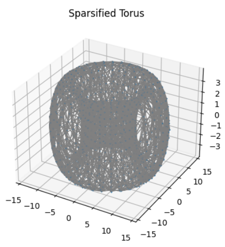


Fig. 6. A sparse torus with $q = 5000$; despite the fact that each vertex may still have roughly five neighbors on average, we find the homology of the structure breaks.

q	1	2	3	4	5
10000	-0.425	0.250	0.605	0.734	0.947
20000	-0.540	0.021	0.534	0.757	0.965
30000	-0.556	0.027	0.570	0.800	0.975

Fig. 7. The correlation between diffusion curvatures at each pair of q and r. Recall the torus has thickness 8, and so for 1000 points, we may imagine that radii less than 1 or 2 result in sparse neighborhoods and are thus not sufficiently informative to begin with.

We can infer from the correlations that the sparsification algorithm is strong at approximating the diffusion curvature of the graph. As in previous sections, we see that this is a global property being carried over accurately into the new graph. This is fairly in line with our intuition about sparsification. While we will likely not preserve exact local structure well, more and more global structures are preserved the more precise our estimate gets. This corresponds precisely with expanding the radius.

V. Conclusion

We have examined the performance of Spielman & Srivastava's algorithm for graph sparsification. We have empirically shown that indeed, effective resistances do act as a reasonable metric for selecting edges in a randomized algorithm. Not only that, but by demonstrating the performance of large samples of edges in comparing the spectrum, we have showed that the randomized embeddings are effective as well. We then show that sparsification allows us not only to approximate graphs, but to, in some sense, denoise the edges of a graph. However, we show that this mode of sparsification is not suitable for removing redundant edges from simplicical complexes for the purpose of toplogical data analysis. In short, this is a effective method of spectral sparsification, and thus spectral approximation, which is suitable for characterizing global geometry.

References

[1] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[2] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, oct 2008.

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

[4] Kenneth M Hall. An r-dimensional quadratic placement algorithm. *Management science*, 17(3):219–229, 1970.

[5] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

[6] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[7] Daniel Spielman. Spectral graph theory. *Combinatorial scientific computing*, 18, 2012.

[8] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 563–568, 2008.

[9] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90, 2004.