SGL: Spectral Graph Learning from Measurements

Zhuo Feng
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, NJ, USA
zhuo.feng@stevens.edu

2021 58th ACM/IEEE Design Automation Conference (DAC) | 978-1-6654-3274-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/DAC18074.2021.9586124

***Abstract*—This work introduces a highly-scalable spectral graph densification framework for learning resistor networks with linear measurements, such as node voltages and currents. We prove that given $O(\log N)$ pairs of voltage and current measurements, it is possible to recover ultra-sparse N-node resistor networks which can well preserve the effective resistance distances on the graph. In addition, the learned graphs also preserve the structural (spectral) properties of the original graph, which can potentially be leveraged in many circuit design and optimization tasks. We show that the proposed graph learning approach is equivalent to solving the classical graphical Lasso problems with Laplacian-like precision matrices. Through extensive experiments for a variety of real-world test cases, we show that the proposed approach is highly scalable for learning ultra-sparse resistor networks without sacrificing solution quality.**

***Index Terms*—spectral graph theory, graph Laplacian estimation, graphical Lasso, convex optimization, resistor networks**

I. Introduction

Recent years have seen a surge of interest in machine learning on graphs, with the goal of encoding high-dimensional data associated with nodes, edges, or (sub)graphs into low-dimensional vector representations that well preserve the original graph structural (manifold) information. Graph learning techniques have shown promising results for various important applications such as vertex (data) classification, link prediction (recommendation systems), community detection, drug discovery, and electronic design automation (EDA) [12].

Modern graph learning involves the following two key tasks: (1) graph topology learning for converting high-dimensional node feature (attribute) data into a graph representation, and (2) graph embedding for converting graph-structured data (e.g. graph topology and node features) into low-dimensional vector representations. For example, an increasingly popular approach for analysing a data set in high-dimensional space (e.g. images of hand-written digits) is to first construct a graph connecting all the data points according to their similarities measured in distances in certain metric space [2]; next, graph embedding techniques are used to compute a low-dimensional vector representation of each data point (graph vertex), so that existing downstream machine learning or data mining algorithms can be conveniently applied.

However, even the state-of-the-art graph learning methods [2], [3] do not scale comfortably to large data sets due to their high algorithm complexity. For example, recent graph learning methods based on Laplacian matrix estimation [2], [3] have shown very promising performance. However, solving the required convex optimization problem has a time complexity of $O(N^2)$ per iteration for N data points, which limits the application of these techniques to only very small data sets (e.g., with up to a few thousands data points).

For the first time, this paper introduces a spectral method for learning resistor networks from linear voltage and current measurements. Our approach is based on a scalable spectral graph densification algorithm (SGL) for estimation of attractive Gaussian Markov Random Fields (GMRFs). The proposed SGL algorithm can efficiently solve the graphical Lasso problem [4] with a Laplacian-like precision matrix by iteratively including the most influential edges to dramatically reduce spectral embedding distortions. A unique property of the learned graphs is that the spectral embedding or effective-resistance distances on the constructed graph will encode the similarities between the original input data points (node voltage measurements). To achieve high efficiency, SGL exploits a scalable spectral graph embedding scheme, which allows each iteration to be completed in $O(N \log N)$ time, whereas existing state-of-the-art methods [2], [3] require at least $O(N^2)$ time for each iteration. Our analysis for sample complexity shows that by leveraging the SGL algorithm it is possible to accurately estimate a sparse resistor network with only $O(\log N)$ voltage (and current) measurements (vectors).

The rest of this paper is organized as follows. Section II introduces the proposed spectral graph learning (SGL) framework is described in detail, which also includes the sample and algorithm complexity analysis. Section III demonstrates extensive experimental results for learning a variety of real-world, large-scale graph problems, which is followed by the conclusion of this work in Section IV.

II. SGL: A Spectral Learning Approach

Suppose we are given M linear measurements of N-dimensional voltage and current vectors stored in data matrices $X \in \mathbb{R}^{N\times M}$ and $Y \in \mathbb{R}^{N\times M}$. The i-th column vector $X(:,i)$ corresponds to a voltage response (graph signal) vector due to the i-th current excitation vector $Y(:,i)$. Motivated by recent graph learning research [2], we propose a scalable spectral method (SGL) for graph Laplacian matrix estimation by exploiting the voltage (X) and current (Y) measurements.

A. Graph Learning via Laplacian Estimation

Similar to the graphical Lasso problem [4], the recent graph signal processing (GSP) based Laplacian estimation methods [2] aim to learn graph structures such that graph signals will vary smoothly across connected neighboring nodes [2]. To quantify the smoothness of a graph signal vector x over a undirected graph $G = (V, E, w)$, the following Laplacian quadratic form can been adopted:

$$x^\top Lx = \sum_{(s,t)\in E} w_{s,t}(x(s) - x(t))^2, \quad (1)$$

where $L = D - W$ denotes the graph Laplacian matrix, $w_{s,t} = W(s,t)$ denotes the weight for edge (s,t), while D and W denote the degree and the weighted adjacency matrices, respectively. The GSP-based graph learning targets the following convex optimization task [2]:

$$\max_{\Theta}: \ F = \log\det(\Theta) - \frac{1}{M}Tr(X^\top \Theta X) - \beta\|\Theta\|_1, \quad (2)$$

where $\Theta = L + \frac{I}{\sigma^2}$, L denotes the set of valid graph Laplacian matrices, $Tr(\bullet)$ denotes the matrix trace, I denotes the identity matrix, and $\sigma^2 > 0$ denotes prior feature variance. In addition, $\|\bullet\|$ denotes the entry-wise ℓ_1 norm, so $\beta\|\Theta\|_1$ becomes the sparsity promoting regularization term. Since $\Theta = L + \frac{I}{\sigma^2}$ is a symmetric and positive definite (PSD) matrix (or M matrix) with non-positive off-diagonal entries, this formulation will lead to the estimation of attractive GMRFs [2], [9].

B. Gradient Estimation via Sensitivity Analysis

We can express the graph Laplacian matrix as follows

$$L = \sum_{(s,t)\in E} w_{s,t} e_{s,t} e_{s,t}^\top \quad (3)$$

where $e_s \in \mathbb{R}^N$ denotes the standard basis vector with all zero entries except for the s-th entry being 1, and $e_{s,t} = e_s - e_t$. Substituting (3) into the objective function F in (2), and taking the partial derivative with respect to $w_{s,t}$ leads to:

$$\frac{\partial F}{\partial w_{s,t}} = \sum_{i=1}^{N} \frac{1}{\lambda_i + 1/\sigma^2} \frac{\partial \lambda_i}{\partial w_{s,t}} - \frac{1}{M}\|X^\top e_{s,t}\|_2^2 - 4\beta, \quad (4)$$

where the eigenvectors corresponding to the ascending eigenvalues λ_i are denoted by u_i for $i = 1, ..., N$, which satisfies:

$$Lu_i = \lambda_i u_i. \quad (5)$$

Note that the last two terms in (4) both imply constraints on graph sparsity: including more edges will result in a greater trace $Tr(X^\top \Theta X)$. Consequently, we can safely choose to set $\beta = 0$ in the rest of this work, which will not impact the ranking of influential edges and thus the solution quality of the proposed SGL algorithm.

Theorem II.1. *The spectral perturbation $\delta\lambda_i$ due to the inclusion of a candidate edge (s,t) can be estimated by:*

$$\delta\lambda_i = \delta w_{s,t}\left(u_i^\top e_{s,t}\right)^2. \quad (6)$$

Proof. Consider the following spectral perturbation analysis:

$$(L + \delta L)(u_i + \delta u_i) = (\lambda_i + \delta\lambda_i)(u_i + \delta u_i), \quad (7)$$

where a perturbation δL that includes a new edge connection is applied to L, resulting in perturbed eigenvalues and eigenvectors $\lambda_i + \delta\lambda_i$ and $u_i + \delta u_i$ for $i = 1, ..., N$, respectively. Keeping only the first-order terms leads to:

$$L\delta u_i + \delta L u_i = \lambda_i \delta u_i + \delta\lambda_i u_i. \quad (8)$$

Write δu_i in terms of the original eigenvectors u_i for for $i = 1, ..., N$:

$$\delta u_i = \sum_{i=1}^{N} \alpha_i u_i. \quad (9)$$

Substituting (9) into (8) leads to:

$$L\sum_{i=1}^{N} \alpha_i u_i + \delta L u_i = \lambda_i \sum_{i=1}^{N} \alpha_i u_i + \delta\lambda_i u_i. \quad (10)$$

Multiplying $u_i^\top$ to both sides of (10) leads to:

$$\delta\lambda_i = \delta w_{s,t}\left(u_i^\top e_{s,t}\right)^2. \quad (11)$$

□

For a connected graph, we construct the following subspace matrix for spectral graph embedding with the first $r - 1$ nontrivial Laplacian eigenvectors

$$U_r = \left[\frac{u_2}{\sqrt{\lambda_2 + 1/\sigma^2}}, ..., \frac{u_r}{\sqrt{\lambda_r + 1/\sigma^2}}\right]. \quad (12)$$

According to Theorem II.1, (4) can be approximated as:

$$s_{s,t} = \frac{\partial F}{\partial w_{s,t}} \approx \|U_r^\top e_{s,t}\|_2^2 - \frac{1}{M}\|X^\top e_{s,t}\|_2^2 = z_{s,t}^{emb} - \frac{1}{M} z_{s,t}^{data}, \quad (13)$$

where $z_{s,t}^{emb} = \|U_r^\top e_{s,t}\|_2^2$ and $z_{s,t}^{data} = \|X^\top e_{s,t}\|_2^2$ denote the ℓ_2 distances in the spectral embedding space and the data (voltage measurement) vector space, respectively. The partial derivative $(s_{s,t})$ in (13) can be regarded as each edge's sensitivity that can be leveraged for solving the optimization task in (2) using gradient based methods, such as the general stagewise algorithm for group-structured learning [11].

C. Convergence Analysis of the SGL Algorithm

If we define the spectral embedding distortion $\eta_{s,t}$ of an edge (s,t) to be:

$$\eta_{s,t} = M\frac{z_{s,t}^{emb}}{z_{s,t}^{data}}. \quad (14)$$

Since $z_{s,t}^{emb}$ is equivalent to the effective resistance $R_{s,t}^{\mathit{eff}}$ on the graph when $\sigma^2 \to +\infty$ and $r \to N$, we can rewrite the spectral embedding distortion as

$$\eta_{s,t} = w_{s,t} R_{s,t}^{\mathit{eff}}, \quad (15)$$

where $w_{s,t} = \frac{M}{z_{s,t}^{data}}$. (15) implies that $\eta_{s,t}$ becomes the edge leverage score for spectral graph sparsification [10]. Prior work shows that every undirected graph has a nearly-linear-sized spectral sparsifier with $O(N \log N)$ edges which can be obtained by sampling each edge with a probability proportional to its edge leverage score [10]; on the other hand, the proposed SGL graph learning framework can be regarded as a **spectral graph densification** procedure that aims to construct a graph with $O(N \log N)$ edges such that the spectral embedding (effective-resistance) distances will encode the ℓ_2 distances between the original data points (voltage measurements). The global (maximum) optimal solution of (2) can be obtained when the maximum edge sensitivity (s_{max}) in (13) becomes zero or equivalently when the maximum spectral embedding distortion (η_{max}) in (14) becomes one.

D. Sample Complexity of the SGL Algorithm

We analyze the required number of voltage vectors (measurements) for accurate graph learning via the SGL approach. Assume that $\sigma^2 \to +\infty$. Denote the ground-truth graph by G_*, and define its edge weight matrix W_* to be a diagonal matrix with $W_*(i,i) = w_i$, and its injection matrix as:

$$B_*(i,p) = \begin{cases} 1 & \text{if } p \text{ is i-th edge's head} \\ -1 & \text{if } p \text{ is i-th edge's tail} \\ 0 & \text{otherwise .} \end{cases} \quad (16)$$

Then the Laplacian matrix of the ground-truth graph G_* in (3) can also be written as $L_* = B_*^\top W_* B_*$. Consequently, the effective resistance $R_*^{eff}(s,t)$ between nodes s and t becomes:

$$R_*^{eff}(s,t) = e_{s,t}^\top L_*^+ e_{s,t} = \|W_*^{\frac{1}{2}} B_* L_*^+ e_{s,t}\|_2^2, \quad (17)$$

where L_*^+ denotes the Moore–Penrose pseudoinverse of L_*. According to the Johnson-Lindenstrauss Lemma, the effective-resistance distance for every pair of nodes satisfies [10]:

$$(1-\epsilon) R_*^{eff}(s,t) \le \|X^\top e_{s,t}\|_2^2 \le (1+\epsilon) R_*^{eff}(s,t), \quad (18)$$

where the voltage measurement matrix $X \in \mathbb{R}^{N \times M}$ is constructed by going through the following steps:

1) Let C be a random $\pm\frac{1}{\sqrt{M}}$ matrix of dimension $M \times |E|$, where $|E|$ denotes the number of edges and $M = 24 \log \frac{N}{\epsilon^2}$ denotes the number of voltage measurements;
2) Obtain $Y = C W_*^{\frac{1}{2}} B_*$, with the i-th row vector denoted by $y_i^\top$;
3) Solve $L_* x_i = y_i$ for all rows in C ($1 \le i \le M$), and construct X using x_i as its i-th column vector.

Consequently, given $M \ge O(\log N/\epsilon^2)$ voltage vectors (measurements) obtained through the above procedure, a $(1 \pm \epsilon)$-approximate effective-resistance distance can be computed by $\tilde{R}_*^{eff}(s,t) = \|X^\top e_{s,t}\|_2^2$ for any pair of nodes (s,t) in the original graph G_*. Consider the following close connection between effective resistances and spectral graph properties:

$$R_{s,t}^{eff} = \|U_N^\top e_{s,t}\|_2^2, \text{ where } U_N = \left[\frac{u_2}{\sqrt{\lambda_2}}, ..., \frac{u_N}{\sqrt{\lambda_N}}\right]. \quad (19)$$

Consequently, using $O(\log N)$ measurements (sample voltage vectors) would be sufficient for SGL to learn an N-node graph for well preserving the original graph spectral properties.

E. Key Steps in the SGL Algorithm

To achieve good efficiency in graph learning that may involve a large number of nodes, the proposed SGL algorithm can iteratively identify and include the most influential edges into the latest graph until no such edges can be found, through the following key steps.

1) Step 1: Initial Graph Construction: (13) implies that by iteratively identifying and adding the most influential edges (with the highest sensitivities) into the latest graph, the graph spectral embedding (or effective-resistance) distance will encode the ℓ_2 distances between the original data vector space (averaged among M measurements). To gain faster convergence of SGL, sparsified k-nearest-neighbor (kNN) graphs [8] can be leveraged as the initial graphs. However, choosing an optimal k value (the number of nearest neighbors) for constructing the kNN graph can still be challenging for general graph learning tasks: choosing a too large k allows well approximating the global structure of the manifold for the original data points (voltage measurements), but will result in a rather dense graph; choosing a too small k may lead to many small isolated graphs, which may slow down the iterations.

Since circuit networks are typically very sparse (e.g. 2D or 3D meshes) in nature, the voltage or current measurements (vectors) usually lie near low-dimensional manifolds, which allows finding a proper k for our graph learning tasks. To achieve a good trade-off between complexity and quality, in SGL the initial graph will be set up through the following steps: **(1)** Construct a connected kNN graph with a relatively small k value (e.g. $5 \le k \le 10$), which will suffice for approximating the global manifold corresponding to the original measurement data; **(2)** Sparsify the kNN graph by extracting a maximum spanning tree (MST) that will serve as the initial graph. Later, SGL will gradually improve the graph by iteratively including the most influential off-tree edges selected from the kNN graph until convergence.

2) Step 2: Spectral Graph Embedding: Spectral graph embedding directly leverages the first few nontrivial eigenvectors for mapping nodes onto low-dimensional space [1]. The eigenvalue decomposition of Laplacian matrix is usually the computational bottleneck in spectral graph embedding, especially for large graphs. To achieve good scalability, we can exploit fast multilevel eigensolvers that allow computing the first few Laplacian eigenvectors in nearly-linear time without loss of accuracy [16].

3) Step 3: Influential Edge Identification: Once the first few Laplacian eigenvectors are available, we can efficiently identify the most influential off-tree edges by looking at each candidate edge's sensitivity score defined in (13). In the proposed SGL approach, each candidate off-tree edge (in the kNN graph) will be sorted according to its edge sensitivity. Only a few most influential edges that have the largest sensitivities computed by (13) will be included into the latest graph. Note that when $r \ll N$, the following inequality holds for any edge (s,t):

$$\|U_r^\top e_{s,t}\|_2^2 = z_{s,t}^{emb} < \|U_N^\top e_{s,t}\|_2^2 \le R^{eff}(s,t), \quad (20)$$

implying that the sensitivities ($s_{s,t}$) computed using the first r eigenvectors will always be smaller than the actual ones. Obviously, using more eigenvectors for spectral embedding will lead to more accurate estimation of edge sensitivities. For typical circuit networks, sensitivities computed using a small number (e.g. $r < 5$) of eigenvectors will suffice for identifying the most influential edges.

4) Step 4: Convergence Checking: In this work, we propose to exploit the maximum edge sensitivities computed by (13) for checking the convergence of SGL iterations. If there exists no additional edge that has a sensitivity greater than a given threshold ($s_{max} \ge tol$), the SGL iterations can be terminated. It should be noted that choosing different tolerance (tol) levels will result in graphs with different densities. For example, choosing a smaller threshold will require more edges to be included so that the resultant spectral embedding distances on the learned graph can more precisely encode the distances between the original data points.

5) Step 5: Spectral Edge Scaling: Assume that σ^2 in (12) approaches $+\infty$ and the normalized input right-hand-side (current) vectors ($Y = [y_1, ..., y_M]$) corresponding to the M

voltage measurements ($X = [x_1, ..., x_M]$) are orthogonal to the all-one vector. Then for each original voltage vector x_i and its corresponding current vector y_i we have:

$$\|x_i\|_2^2 = y_i^\top (L_*^+)^2 y_i, \;\; for \;\; i = 1, ..., M. \tag{21}$$

Next, for each y_i we compute the voltage vector $\tilde{x}_i$ using the estimated Laplacian L obtained via SGL iterations:

$$L\tilde{x}_i = y_i => \|\tilde{x}_i\|_2^2 = y_i^\top (L^+)^2 y_i, \;\; for \;\; i = 1, ..., M. \tag{22}$$

To more precisely match the original graph spectral properties, each edge weight can be adjusted as follows:

$$w_{s,t} = \tilde{w}_{s,t} * \sqrt{\frac{1}{M} \sum_{i=1}^{M} \frac{\|\tilde{x}_i\|_2^2}{\|x_i\|_2^2}}, \tag{23}$$

where $\tilde{w}_{s,t}$ denotes the initial edge weight obtained via the previous SGL iterations. Since solving the ultra-sparse Laplacian matrix L can be accomplished in nearly linear time [7], [14], the proposed scaling scheme is highly efficient.

F. Algorithm Flow and Complexity

The detailed SGL algorithm flow has been shown in Algorithm 1. All the aforementioned steps in SGL can be accomplished in nearly-linear time by leveraging recent high-performance algorithms for kNN graph construction [8], spectral graph embedding for influential edge identification [13], [16], and fast Laplacian solver for edge scaling [7], [14]. Consequently, each SGL iteration can be accomplished in nearly-linear time, whereas the state-of-the-art methods require at least $O(N^2)$ time [2].

Algorithm 1 The SGL Algorithm Flow

Input: The voltage measurement matrix $X \in \mathbb{R}^{N \times M}$, input current measurement matrix $Y \in \mathbb{R}^{N \times M}$, k for initial kNN graph construction, r for constructing the projection matrix in (12), the maximum edge sensitivity tolerance (tol), and the edge sampling ratio ($0 < \beta \leq 1$). **Output:** The learned graph $G = (V, E, w)$.

```
1:  Construct a kNN graph $G_o = (V, E_o, w_o)$ based on $X$.
2:  Extract an MST subgraph $T$ from $G_o$.
3:  Assign $G = T = (V, E, w)$ as the initial graph.
4:  while $s_{max} \geq tol$ do
5:      Compute the projection matrix $U_r$ with (12) for the latest graph $G$.
6:      Sort off-tree edges $(s, t) \in E_o \setminus E$ according to their sensitivities computed by $s_{s,t} = \frac{\partial F}{\partial w_{s,t}}$ using (13).
7:      Include an off-tree edge $(s, t)$ into $G$ if its $s_{s,t} > tol$ and it has been ranked among the top $\lceil N\beta \rceil$ edges.
8:      Record the maximum edge sensitivity $s_{max}$.
9:  end while
10: Do spectral edge scaling using $X$ and $Y$ via (21)-(23);
11: Return the learned graph $G$.
```

III. Experimental Results

The proposed SGL algorithm has been implemented in Matlab. The test cases in this paper have been selected from a great variety of matrices that have been used in circuit simulation and finite element analysis problems. Since the prior state-of-the-art graph learning algorithms [2] have been developed based on a standard convex solver [5], the runtime would be excessively long (over many thousands of seconds) even for the smallest test case ($|V| = 4,253$) reported in this paper. Therefore, we will only compare with the graph construction method based on the standard kNN algorithm. All of our experiments have been conducted using a single CPU core of a computing system with a 3.4 GHz Quad-Core Intel Core i5 CPU and 24 GB memory.

A. Experimental Setup

To generate the voltage and current measurement samples, the following procedure has been applied: (1) we first randomly generate M current source vectors with each element sampled using a standard normal distribution; (2) each current vector will be normalized and orthogonal to the all-one vector; (3) M voltage vector measurements will be obtained by solving the original graph Laplacian matrix with the M current vectors as the (right-hand-side) input vectors; (4) the voltage and current vectors will be stored in matrices $X = [x_1, ..., x_M]$ and $Y = [y_1, ..., y_M] \in \mathbb{R}^{N \times M}$, respectively, which will be used as the input data of the proposed SGL algorithm. By default, $M = 50$ is used for generating the voltage and current measurements. We choose $k = 5$ for constructing the kNN graph for all test cases. We set $r = 5$ for constructing the projection matrix in (12). The edge sampling ratio $\beta = 10^{-3}$ has been used. The SGL iterations will be terminated if $s_{max} < tol = 10^{-12}$. When approximately computing the objective function value (2), the first 50 nonzero Laplacian eigenvalues are used.

To clearly visualize each graph, the spectral graph drawing technique has been adopted [6]: when creating the 2D graph layouts, each entry of the first two nontrivial Laplacian eigenvectors (u_2, u_3) corresponds to the x and y coordinates of each node (data point), respectively. We assign the nodes with the same color if they belong to the same node cluster determined by spectral graph clustering [15].

B. Comprehensive Results for Graph Learning

a) Algorithm Convergence: As shown in Figure 1, for the "2D mesh" graph ($|V| = 10,000$, $|E| = 20,000$) learning task, SGL requires about 40 iterations to converge to $s_{max} \leq 10^{-12}$ when starting from an initial MST of a 5NN graph.

b) Comparison with kNN Graph: As shown in Figure 2, for the "fe_4elt2" graph ($|V| = 11,143$, $|E| = 32,818$) learning task, SGL converges in about 90 iterations when starting from an initial MST of a 5NN graph. For the 5NN graph, we do the same edge scaling using (21)-(23). As shown in Figures 2 and 3, the SGL-learned graph achieves a more optimal objective function value and a much better spectral approximation than the 5NN graph. As observed, the SGL-learned graph has a density similar to a spanning tree, which is much sparser than the 5NN graph.

c) Learning Circuit Networks: As shown in Figures 4 and 5 for the "airfoil" ($|V| = 4,253$, $|E| = 12,289$) and the "G2_circuit" ($|V| = 150,102$, $|E| = 288,286$) graphs, SGL can consistently learn ultra-sparse graphs which are slightly denser than spanning trees while preserving the key graph spectral properties. In Figure 6, we observe highly correlated results when comparing the effective resistances computed on the original graphs with the graphs learned by SGL.

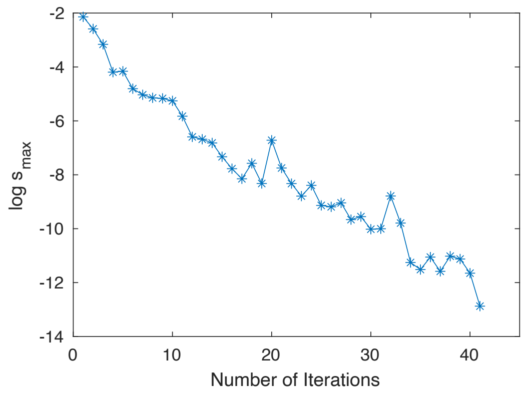

Figure 1. The decreasing maximum sensitivities ("2D mesh" graph)

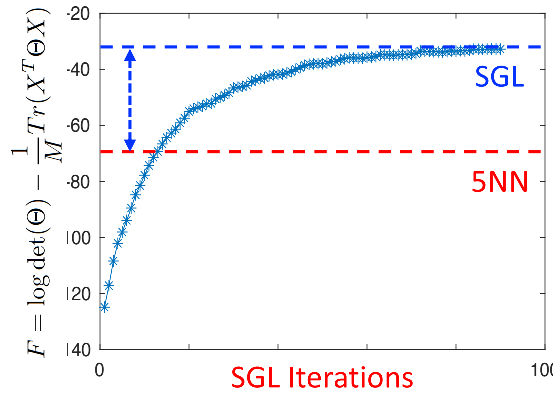

Figure 2. The objective function values ("fe_4elt2" graph)

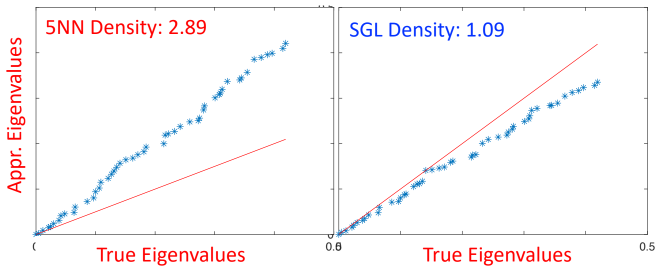

Figure 3. The comparison with a 5NN graph ("fe_4elt2" graph)

d) Learning with Noisy Measurements: We show the results of the "2D mesh" graph learning with noisy voltage measurements. For each SGL graph learning task, each input voltage measurement (vector) $\tilde{x}$ will be computed by: $\tilde{x} = x + \zeta\|x\|_2\epsilon$, where ϵ denotes a normalized Gaussian noise vector, and ζ denotes the noise level. As shown in Figure 7, the increasing noise levels will result in worse approximations of the original spectral properties. It is also observed that even with a very significant noise level of $\zeta = 0.5$, the graph learned by the proposed SGL algorithm can still preserve the first few Laplacian eigenvalues that are key to the graph structural (global) properties.

e) Sample Complexity and Runtime Scalability: Figure 8 shows how the sample complexity (number of measurements) may impact the graph learning quality. As observed, with increasing number of samples (measurements), substantially improved approximation of the graph spectral properties can be achieved. In the last, we show the runtime scalability of the proposed SGL algorithm. The runtime includes the total time of Step 2 to Step 5 but does not include the time for Step 1. Note that modern kNN algorithms can achieve highly scalable runtime performance [8].

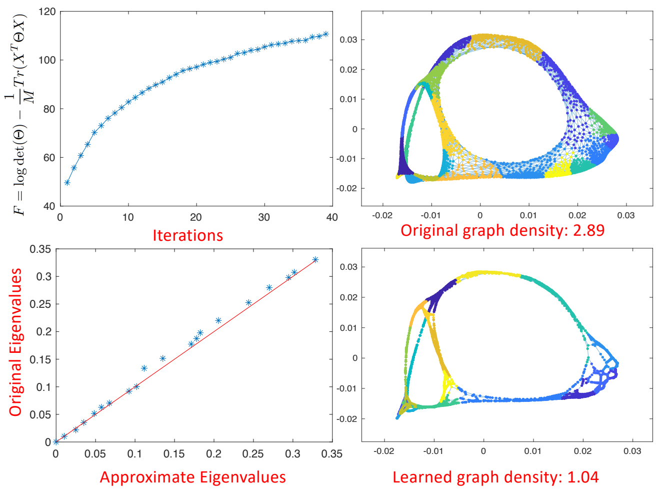

Figure 4. The results for learning the "airfoil" graph

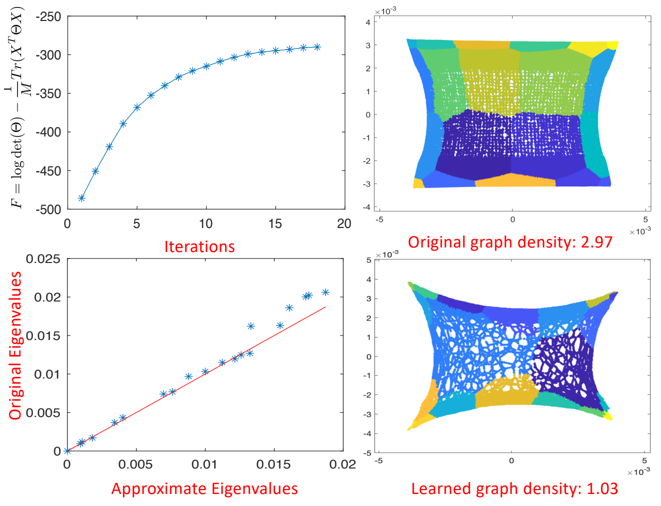

Figure 5. The results for learning the "G2_circuit" graph

IV. Conclusions

This work proposes a spectral algorithm (SGL) for learning resistor networks from linear voltage and current measurements. Our approach iteratively identifies and includes the most influential edges to the latest graph. We show that the proposed graph learning approach is equivalent to solving the classical graphical Lasso problems with Laplacian-like precision matrices. A unique feature of SGL is that the learned graphs will have spectral embedding or effective-resistance distances encoding the similarities between the original input data points (node voltages). To achieve high efficiency, SGL exploits a scalable spectral embedding scheme to allow each

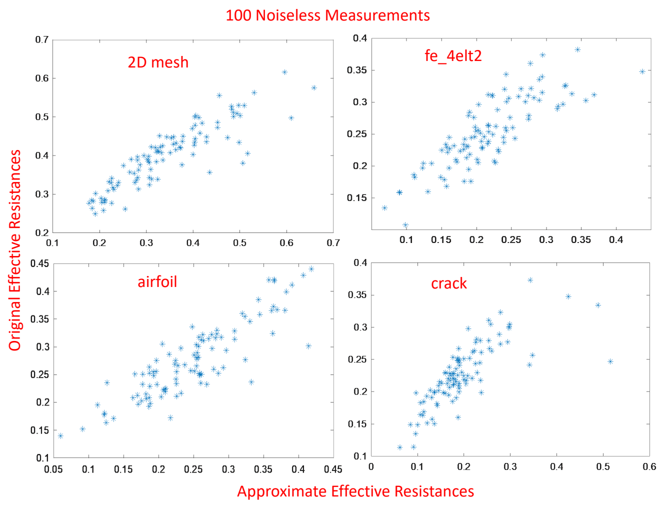

Figure 6. The effective resistances correlations (scatter plots)

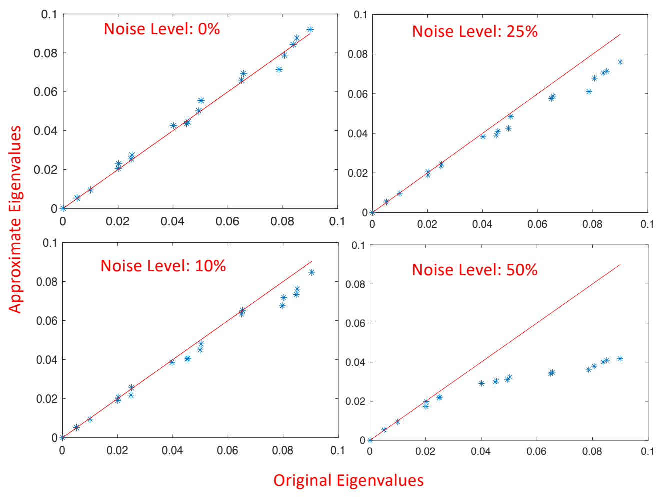

Figure 7. The graphs learned with noises ("2D mesh" graph)

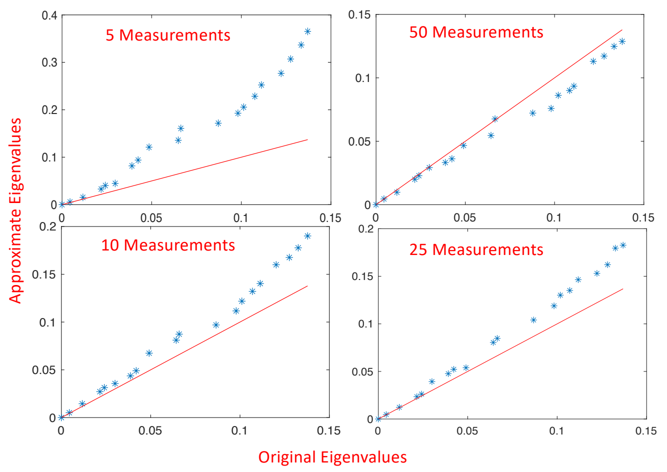

Figure 8. The effect of the number of measurements ("fe_4elt2" graph)

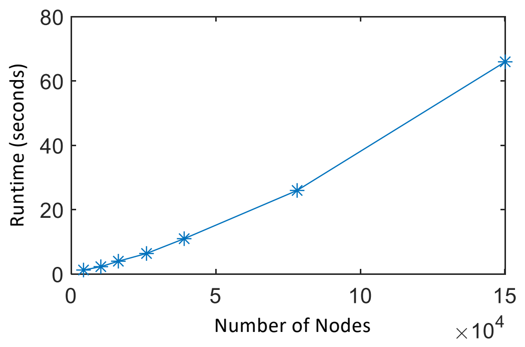

Figure 9. The runtime scalability of the SGL algorithm

iteration to be completed in $O(N \log N)$ time, whereas existing state-of-the-art methods require at least $O(N^2)$ time for each iteration. We also provide a sample complexity analysis showing that it is possible to accurately recover a resistor network with only $O(\log N)$ voltage measurements (vectors).

V. Acknowledgments

This work is supported in part by the National Science Foundation under Grants CCF-2041519 (CAREER), CCF-2021309 (SHF), and CCF-2011412 (SHF).

References

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

[2] X. Dong, D. Thanou, M. Rabbat, and P. Frossard. Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine*, 36(3):44–63, 2019.

[3] H. E. Egilmez, E. Pavez, and A. Ortega. Graph learning from data under laplacian and structural constraints. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):825–841, 2017.

[4] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[5] M. Grant, S. Boyd, and Y. Ye. Cvx: Matlab software for disciplined convex programming, 2009.

[6] Y. Koren. On spectral graph drawing. In *International Computing and Combinatorics Conference*, pages 496–508. Springer, 2003.

[7] I. Koutis, G. Miller, and R. Peng. Approaching Optimality for Solving SDD Linear Systems. In *Proc. IEEE FOCS*, pages 235–244, 2010.

[8] Y. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[9] M. Slawski and M. Hein. Estimation of positive definite m-matrices and structure learning for attractive gaussian markov random fields. *Linear Algebra and its Applications*, 473:145–179, 2015.

[10] D. Spielman and N. Srivastava. Graph Sparsification by Effective Resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[11] R. J. Tibshirani. A general framework for fast stagewise algorithms. *The Journal of Machine Learning Research*, 16(1):2543–2588, 2015.

[12] H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H.-S. Lee, and S. Han. GCN-RL Circuit Designer: Transferable Transistor Sizing with Graph Neural Networks and Reinforcement Learning. *arXiv preprint arXiv:2005.00406*, 2020.

[13] Z. Zhao and Z. Feng. Effective-resistance preserving spectral reduction of graphs. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, pages 109:1–109:6, New York, NY, USA, 2019. ACM.

[14] Z. Zhao, Y. Wang, and Z. Feng. SAMG: Sparsified Graph Theoretic Algebraic Multigrid for Solving Large Symmetric Diagonally Dominant (SDD) Matrices. In *Proceedings of the 36th International Conference on Computer-Aided Design (ICCAD)*. ACM, 2017.

[15] Z. Zhao, Y. Wang, and Z. Feng. Nearly-linear time spectral graph reduction for scalable graph partitioning and data visualization. *arXiv preprint arXiv:1812.08942*, 2018.

[16] Z. Zhao, Y. Zhang, and Z. Feng. Towards scalable spectral embedding and data visualization via spectral coarsening. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 869–877, 2021.