Scalable Graph Topology Learning via Spectral Densification

Yongyu Wang
Michigan Technological University
Houghton, Michigan, USA
yongyuw@mtu.edu

Zhiqiang Zhao
Stevens Institute of Technology
Hoboken, New Jersey, USA
zzhao76@stevens.edu

Zhuo Feng
Stevens Institute of Technology
Hoboken, New Jersey, USA
Zhuo.Feng@stevens.edu

ABSTRACT

Graph learning plays an important role in many data mining and machine learning tasks, such as manifold learning, data representation and analysis, dimensionality reduction, data clustering, and visualization, etc. In this work, we introduce a highly-scalable spectral graph densification approach (GRASPEL) for graph topology learning from data. By limiting the precision matrix to be a graph-Laplacian-like matrix, our approach aims to learn sparse undirected graphs from potentially high-dimensional input data. A very unique property of the graphs learned by GRASPEL is that the spectral embedding (or approximate effective-resistance) distances on the graph will encode the similarities between the original input data points. By leveraging high-performance spectral methods, sparse yet spectrally-robust graphs can be learned by identifying and including the most spectrally-critical edges into the graph. Compared with prior state-of-the-art graph learning approaches, GRASPEL is more scalable and allows substantially improving computing efficiency and solution quality of a variety of data mining and machine learning applications, such as manifold learning, spectral clustering (SC), and dimensionality reduction (DR).

CCS CONCEPTS

• **Computing methodologies** → **Spectral methods**; **Learning paradigms**.

KEYWORDS

graph topology learning; spectral graph theory; spectral clustering; dimensionality reduction

ACM Reference Format:
Yongyu Wang, Zhiqiang Zhao, and Zhuo Feng. 2022. Scalable Graph Topology Learning via Spectral Densification. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3488560.3498480

1 INTRODUCTION

Graph learning is playing increasingly important roles in many machine learning and data mining applications. For example, a key step of many existing machine learning methods requires converting potentially high-dimensional data sets into graph representations: it is a common practice to represent each (high-dimensional) data point as a node, and assign each edge a weight to encode the similarity between the two nodes (data points). The constructed graphs can be efficiently leveraged to represent the underlying structure of a data set or the relationship between data points [8, 15, 17]. However, how to learn meaningful graphs from large data set at scale still remains a challenging problem.

Several recent graph learning methods leverage emerging graph signal processing (GSP) techniques for estimating sparse graph Laplacians, which show very promising results [4–6, 10]. For example, [6] addresses the graph learning problem by restricting the precision matrix to be a graph Laplacian and maximizing a posterior estimation of attractive Gaussian Markov Random Field (GMRF) [1], while an ℓ_1-regularization term is used to promote graph sparsity; [22] provides an error analysis for inferring sparse graphs from smooth signals; [10] leverages approximate nearest-neighbor (ANN) graphs to reduce the number of variables for optimization; [12] introduces a graph Laplacian learning method by imposing Laplacian spectral constraints.

However, even the state-of-the-art Laplacian estimation methods for graph learning do not scale well for large data set due to their extremely high algorithm complexity. For example, solving the optimization problem for Laplacian estimation in [4–6, 9] requires $O(N^2)$ time complexity per iteration for N data entities and nontrivial parameters tuning for controlling graph sparsity which limits their applications to only very small data sets (e. g. with up to a few thousands of data points); the method introduced in [2] leverages Isomap manifold embedding [29] for graph construction, which requires $O(N^3)$ time for manifold construction and thus does not scale to large data set; the latest graph learning approach [10] takes advantages of ANN graphs, but it still runs very slowly for large data sets; the Laplacian estimation method with spectral constraints requires a good graph structure to be provided in advance [12], which otherwise can be very costly when going through exhaustive graph structure searches.

This work introduces a *spectral graph densification* approach (GRASPEL [2]) for learning sparse graphs from data by leveraging spectral graph algorithms. GRASPEL has a close connection with prior GSP-based Laplacian estimation methods [4–6, 9, 10] and graphical Lasso [7]. By treating M-dimensional data points as M graph signals, GRASPEL allows efficiently solving a convex problem by iterative identifying and including the most spectrally-critical candidate edges into the latest graph leveraging recent nearly-linear time spectral methods [33]. Compared with prior spectral graph sparsification algorithms [26] that target pruning edges from a given graph while preserving key graph spectral

WSDM '22, February 21–25, 2022, Tempe, AZ, USA

ACM ISBN 978-1-4503-9132-0/22/02…$15.00
https://doi.org/10.1145/3488560.3498480

[1] If the precision matrix of a GMRF is an M-matrix with all non-negative off-diagonal elements, we call it an attractive GMRF [5, 25].
[2] The source code is available at: https://github.com/Feng-Research/GRASPEL

properties, GRASPEL aims at iteratively densifying graphs such that the learned graphs will have spectral embedding (or effective-resistance) distances encoding the similarities between the original input data points. Comparing with state-of-the-art graph construction methods, GRASPEL is more scalable for estimation of attractive Gaussian Markov Random Fields (GMRFs) even for very large data set. We summarize the contribution of this work as follows:

• We propose a spectral graph densification approach (GRASPEL) that allows efficient estimation of attractive Gaussian Markov Random Fields (GMRFs) leveraging the latest spectral graph theory. The key to achieving high efficiency is a spectral embedding scheme for finding spectrally-critical edges, allowing each GRASPEL iteration to be completed in $O(N \log N)$ instead of $O(N^2)$ time.

• We introduce a novel convergence criterion for GRASPEL iterations based on graph spectral stability: when the maximum embedding distortion becomes relatively small, or equivalently the graph spectrum becomes sufficiently stable, GRASPEL iterations can be terminated.

• Our experiment results show that the graphs learned using GRASPEL can lead to more efficient and accurate spectral clustering (SC) as well as dimensionality reduction (DR).

2 BACKGROUND

Given M observations on N data entities in a data matrix $X = [x_1, ..., x_M] \in \mathbb{R}^{N \times M}$, each column vector of X can be considered as a signal on a graph. For example, the USPS data set including 9,298 images of handwritten digits with each image having 256 pixels will result in a feature matrix $X \in \mathbb{R}^{N \times M}$ with $N = 9,298$ and $M = 256$. The recent GSP-based graph learning methods [4] estimate graph Laplacians from X for achieving the following desired characteristics:

Smoothness of graph signals. The graph signals corresponding to the real-world data should be sufficiently smooth on the learned graph structure: the signal values will only change gradually across connected neighboring nodes. The smoothness of a signal x over an undirected graph $G = (V, E, w)$ can be measured with the following Laplacian quadratic form:

$$x^\top Lx = \sum_{(p,q) \in E} w_{p,q}(x(p) - x(q))^2, \tag{1}$$

where $w_{p,q}$ denotes the weight of edge (p, q), $L = D - W$ denotes the Laplacian, D denotes the diagonal (degree) matrix, and W denotes the adjacency matrix of G, respectively. The smaller value of (1) indicates the smoother signals across the edges in the graph. The smoothness (Q) of a set of signals X over graph G is computed using the matrix trace [9] $Q(X, L) = Tr(X^\top LX)$, where $Tr(\bullet)$ denotes the matrix trace.

Sparsity of the estimated graph. Graph sparsity is another critical consideration in graph learning. One of the most important motivations of learning a graph is to use it for downstream data mining or machine learning tasks. Therefore, desired graph learning algorithms should allow better capturing and understanding the global structure (manifold) of the data set, while producing sufficiently sparse graphs that can be easily stored and efficiently manipulated in the downstream algorithms, such as graph clustering, partitioning, dimension reduction, data visualization, etc.

Prior methods. Consider an N-dimensional random vector x following a multivariate Gaussian distribution $x \sim N(0, \Sigma)$ with probability density:

$$f(x) = \frac{\exp\left(-\frac{1}{2}x^\top \Sigma^{-1} x\right)}{(2\pi)^{N/2} \det(\Sigma)^{(1/2)}} \propto \det(\Theta)^{1/2} \exp\left(-\frac{1}{2}x^\top \Theta x\right), \tag{2}$$

where $\Sigma = \mathbb{E}[xx^\top] > 0$ denotes the covariance matrix, and $\Theta = \Sigma^{-1}$ denotes the precision matrix (inverse covariance matrix). Prior graph topology learning methods aim at estimating sparse precision matrix Θ from potentially high-dimensional input data:

(A) The graphical Lasso method aims at estimating a sparse precision matrix Θ using the following convex optimization based on maximum likelihood estimation of $f(x)$ [7]:

$$\max_{\Theta} : \log\det(\Theta) - Tr(\Theta S) - \beta \|\Theta\|_1, \tag{3}$$

where Θ denotes a non-negative definite precision matrix, S denotes a sample covariance matrix, and β denotes a regularization parameter. The first two terms together can be interpreted as the log-likelihood under a Gaussian Markov Random Field (GMRF). $\|\bullet\|$ denotes the entry-wise ℓ_1 norm, so $\beta\|\Theta\|_1$ becomes the sparsity promoting regularization term. This model learns the graph structure by maximizing the penalized log-likelihood. When the sample covariance matrix S is obtained from M i.i.d (independent and identically distributed) samples $X = [x_1, ..., x_M]$ where $X \sim N(0, S)$ has an N-dimensional Gaussian distribution with zero mean, each element in the precision matrix $\Theta_{i,j}$ encodes the conditional dependence between variables X_i and X_j. For example, $\Theta_{i,j} = 0$ implies that the corresponding variables X_i and X_j are conditionally independent, given the rest.

(B) The GSP-based Laplacian estimation methods have been recently introduced for more efficiently solving the following convex problem [5, 13]:

$$\max_{\Theta} : F(\Theta) = \log\det(\Theta) - \frac{1}{M} Tr(X^\top \Theta X) - \beta \|\Theta\|_1, \tag{4}$$

where $\Theta = L + \frac{1}{\sigma^2} I$, L denotes the set of valid graph Laplacian matrices, I denotes the identity matrix, and $\sigma^2 > 0$ denotes prior feature variance. It can be shown that the three terms in (4) are corresponding to $\log\det(\Theta)$, $Tr(\Theta S)$ and $\beta\|\Theta\|_1$ in (3), respectively. When each column vector in the data matrix X [3] is treated as a graph signal vector, there is a close connection between formulation (4) and the graphical Lasso problem. Since $\Theta = L + \frac{1}{\sigma^2} I$ corresponds to symmetric and positive definite (PSD) matrices (or M matrices) with non-positive off-diagonal entries, this formulation will lead to the estimation of attractive GMRFs [5, 25]. In case X is non-Gaussian, (4) can be understood as Laplacian estimation based on minimizing the Bregman divergence between positive definite matrices induced by the function $\Theta \mapsto -\log\det(\Theta)$ [25].

Express the Laplacian matrix of an undirected graph $G = (V, E, w)$ as follows:

$$L = \sum_{(p,q) \in E} w_{p,q} e_{p,q} e_{p,q}^\top \tag{5}$$

[3] For each of the N row vectors $X(i,:) \in \mathbb{R}^{1 \times M}$ where $i = 1, ..., N$, the following two-step data pre-processing will be performed: **(1)** $X(i,:) = X(i,:) - \mu_i$, where μ_i denotes the sample mean of $X(i,:)$; **(2)** Feature normalization $X = \sqrt{M}X/\|X\|_2$.

where $e_p \in \mathbb{R}^N$ denotes the standard basis vector with all zero entries except for the p-th entry being 1, and $e_{p,q} = e_p - e_q$. The ascending Laplacian eigenvalues and the corresponding eigenvectors are denoted by λ_i and u_i for $i = 1, ..., N$, respectively, which satisfy:

$$Lu_i = \lambda_i u_i. \tag{6}$$

Rewrite the objective function in (4) as follows:

$$F = \log\det\left(L + \frac{1}{\sigma^2}I\right) - \frac{1}{M}Tr(X^\top LX) - \frac{Tr(X^\top X)}{M\sigma^2} - \beta\|L\|_1. \tag{7}$$

3 A SPECTRAL LEARNING APPROACH

Taking the partial derivative with respect to the weight $w_{p,q}$ of edge (p, q) leads to:

$$\frac{\partial F}{\partial w_{p,q}} = \sum_{i=2}^{N} \frac{1}{\lambda_i + 1/\sigma^2}\frac{\partial \lambda_i}{\partial w_{p,q}} - \frac{1}{M}\|X^\top e_{p,q}\|_2^2 - 4\beta, \tag{8}$$

Since the last two terms in (8) are all fixed (constant) values for a given data matrix X where β can be considered as an additional offset added to all data pairs (candidate edges), we can drop the third term by simply setting $\beta = 0$, which will not impact the ranking of candidate edges in graph learning. The above simplification implies the second term alone will effectively penalize graph density for estimating Laplacian-like precision matrix: including more edges will result in a greater trace $Tr(X^\top \Theta X)$.

Define the following **eigensubspace matrix** $U_r \in \mathbb{R}^{N\times(r-1)}$ for spectral graph embedding:

$$U_r = \left[\frac{u_2}{\sqrt{\lambda_2 + 1/\sigma^2}}, ..., \frac{u_r}{\sqrt{\lambda_r + 1/\sigma^2}}\right], \tag{9}$$

which includes the first $r-1$ weighted nontrivial Laplacian eigenvectors as its column vectors. Choosing $\beta = 0$ and $r = N$, then the edge **spectral sensitivity** $s_{p,q}$ can be derived as follows:

$$\begin{aligned} s_{p,q} = \frac{\partial F}{\partial w_{p,q}} &= \sum_{i=2}^{N} \frac{1}{\lambda_i + 1/\sigma^2}\frac{\partial \lambda_i}{\partial w_{p,q}} - \frac{1}{M}\|X^\top e_{p,q}\|_2^2 \\ &= z_{p,q}^{emb} - \frac{1}{M}z_{p,q}^{data}, \end{aligned} \tag{10}$$

where $z_{p,q}^{emb} = \|U_N^\top e_{p,q}\|_2^2$ and $z_{p,q}^{data} = \|X^\top e_{p,q}\|_2^2$ denote the ℓ_2 distances in the spectral embedding space and the original data space, respectively. In practice, choosing a small r (e.g. $2 \le r \ll N$) according to the gaps between the first few eigenvalues will suffice.

Remark 1. *When $\sigma^2 \to +\infty$ and $r = N$, $z_{p,q}^{emb}$ becomes the effective-resistance distance $R_{p,q}^{eff} = e_{p,q}^\top L^+ e_{p,q} = \|U_r^\top e_{p,q}\|_2^2 = z_{p,q}^{emb}$ between nodes p and q on the graph, where $L^+ \in \mathbb{R}^{N\times N}$ denotes the Moore–Penrose pseudoinverse of the Laplacian matrix L.*

Define the **embedding distortion** of a candidate edge (p, q) by

$$\eta_{p,q} = M\frac{z_{p,q}^{emb}}{z_{p,q}^{data}}. \tag{11}$$

Also define **spectrally-critical edge** to be a candidate edge (p, q) that has a large embedding distortion $\eta_{p,q}$. Therefore, if an edge candidate has a large embedding distortion, any small perturbation to this edge can change the objective function F significantly.

3.1 Spectral Graph Densification

Consider the $(k+1)$-th GRASPEL iteration for identifying the most spectrally-critical (top) candidate edge to be included into the latest graph $G_k = (V_k, E_k, w_k)$ [4]. Suppose all $O(N^2)$ candidate edges are initialized with very small (near-zero) weights. In each GRASPEL iteration, the top candidate edge with the largest spectral sensitivity will be identified and assigned with a much larger edge weight, which is equivalent to adding the candidate edge into the latest graph. According to (10), including such an edge into G_k will significantly improve the objective function and decrease the spectral embdding distortion. To avoid storing $O(N^2)$ candidate edges, GRASPEL leverages a nearly-linear time spectral densification scheme that has been shown in Figure 1.

Spectral graph sparsification (prior work). Prior work proves that every undirected graph has a sparsified graph with $O(\frac{N\log N}{\epsilon^2})$ edges that can be obtained by sampling each edge (p, q) with a probability (leverage score) p_e proportional to its effective resistance $p_e \propto \frac{R_{p,q}^{eff}}{R_{p,q}} = w_{p,q}R_{p,q}^{eff}$, where $R_{p,q} = 1/w_{p,q}$ denotes the original resistance and $R_{p,q}^{eff}$ denotes the effective resistance. The sparsified Laplacian satisfies the following inequalities [26]:

$$\forall x \in \mathbb{R}^N \ (1-\epsilon)x^\top Lx \le x^\top \tilde{L}x \le (1+\epsilon)x^\top Lx, \tag{12}$$

where L and $\tilde{L}$ denote the original and sparsified graph Laplacians, respectively.

Spectral graph densification (this work). For a candidate edge that has been selected according to its spectral sensitivity during a GRASPEL iteration, by setting its edge weight as $w_{p,q} \propto \frac{1}{z_{p,q}^{data}}$, the spectral embedding distortion $\eta_{p,q} \propto w_{p,q}R_{p,q}^{eff}$ becomes the leverage score for spectral graph sparsification [26]. Therefore, GRASPEL iterations can be considered as a spectral graph densification procedure that aims to include $O(N\log N)$ spectrally-critical edges with large spectral sensitivities (embedding distortions). Moreover, upon convergence the spectral embedding (effective-resistance) distances on the graphs learned by GRASPEL will encode the ℓ_2 distances between the original data points, which is key to manifold learning and dimensionality reduction problems [1, 2].

Convergence analysis. The global optimal solution can be obtained when (10) becomes zero or there exists no edge with $\eta > 1$ for inclusion to the latest graph. Therefore, the GRASPEL iterations can be terminated when the maximum embedding distortion η_{max} becomes small enough (e.g. $\eta_{max} \le 10$), or equivalently, when the graph spectrum become sufficiently stable (The first few eigenvalues/eigenvectors will not be significantly perturbed by adding new candidate edges).

Algorithm complexity. The subspace matrix U_r with $2 \le r \ll N$ can be computed in $O(N\log N)$ time [33], which allows the spectral embedding distortion $\eta_{p,q}$ of each candidate edge (p, q) to be estimated in constant time $O(1)$. Therefore, each top candidate edge can be identified in $O(N\log N)$ time (see details in Section 4), which is much faster than the state-of-the-art graph topology learning methods [4–6, 9, 10] which require at least $O(N^2)$ time in each iteration.

[4] Assume that $|V_k| = |V| = N$ (every data point is connected in G_k).

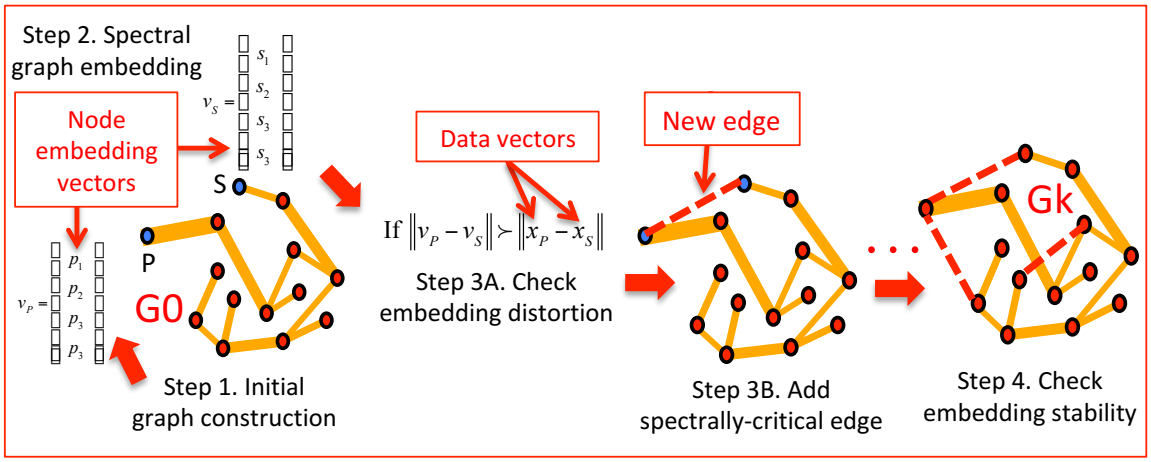

Figure 1: Spectral graph densification in GRASPEL iterations.

4 DETAILED STEPS IN GRASPEL

GRASPEL will iteratively identify and add the top candidate edges into the latest graph so that the spectral embedding distortion can be greatly mitigated, until no such edges can be found (as illustrated in Figure 1). The detailed GRASPEL algorithm flow for graph learning has been described in Algorithm 1, and summarized as follows.

Step 1: Initial graph construction. As aforementioned, (approximate) kNN graphs can be constructed as the initial graph, since they can be obtained efficiently [18], while being able to approximate the local data proximity [23]. However, the optimal k value (the number of nearest neighbors) is usually problem dependent and can be very difficult to find. In this work, GRASPEL will start with creating an (approximate) kNN graph using a relatively small k value (e.g. k = 2 to 5), which will suffice for approximating the local structure of the manifold, and strive to iteratively improve the approximation of the global manifold structure by adding a small portion of spectrally-critical edges through solving the proposed GRASPEL iterations.

Step 2: Spectral graph embedding. Spectral graph embedding directly leverages the first few nontrivial eigenvectors for projecting nodes onto a low-dimensional subspace [1]. The eigenvalue decomposition of Laplacian matrix is usually the computational bottleneck in spectral graph embedding, especially for large graphs [3, 24, 31]. To achieve good scalability, nearly-linear time Laplacian solvers [11] or multilevel Laplacian solvers [33] can be exploited for much faster eigenvector (eigenvalue) computations without loss of accuracy.

Step 3: Spectrally-critical edge identification. Once Laplacian eigenvectors are available for the latest graph, through the following phases GRASPEL will identify spectrally-critical edges by looking at each candidate edge's spectral sensitivity $s_{p,q}$ defined in (10). However, an exhaustive search among all node pairs would require $O(N^2)$ evaluations. To gain much higher efficiency, the following two-phase search strategy has been proposed.

Phase A: candidate edge identification with Fiedler vectors. Our approach for identifying spectrally-critical edges starts with sorting nodes according to the Fiedler vector that can be computed in nearly-linear time leveraging fast Laplacian solvers [11, 27]. This scheme is equivalent to including only the first nontrivial Laplacian eigenvector into U_r (choosing $r = 2$) in (9) for spectral graph embedding. Subsequently, we can search candidate edge connections between the top and bottom few nodes in the 1D sorted node array. According to (10), only a small portion of node pairs with large embedding distances needs to be examined as candidate edges.

Phase B: embedding distortion estimation with multiple eigenvectors. With the first r Laplacian eigenvectors computed in the previous spectral embedding step, each node of the latest graph will be

Algorithm 1 The GRASPEL Algorithm Flow

Input: A data matrix ($X = [x_1, ...x_M] \in \mathbb{R}^{N \times M}$) with N data points in M-dimensional, embedding distortion tolerance ($1 \leq tol$), window size for edge sampling ($0 < \epsilon \leq 50\%$), edge sampling ratio ($0 < \zeta \leq 1$), and the number of edges to be selected in each iteration ($0 < s$). **Output:** The spectrally-learned graph G.

1: Construct an initial 2NN graph G using approximate kNN algorithms.
2: **while** $\eta_{max} \geq tol$ **do**
3: Embed the latest graph G using its Fiedler vector and sort the nodes into a 1D array I_{node};
4: Obtain node set N_{top} (N_{bot}) by including only the top (bottom) $\lceil \epsilon N \rceil$ nodes in I_{node};
5: Sample each of the $\lceil s/\zeta \rceil$ edges by randomly choosing one node from N_{top} and another node from N_{bot};
6: Form an edge set E_{sel} using edges with large distortions ($\eta \geq tol$) and set the largest edge embedding distortion as η_{max}.
7: **if** $|E_{sel}| \geq s$ **then**
8: Add the top s edges from E_{sel} into G;
9: **else**
10: Add all the edges in E_{sel} into G;
11: **end if**
12: **end while**
13: Return the learned graph G.

associated with an r-dimensional embedding vector, which allows the spectral embedding distortion of each candidate (spectrally-critical) edge to be quickly estimated. As aforementioned, the spectral embedding distances computed with the first r eigenvectors can well approximate the effective-resistance distance thus the edge sensitivity in the proposed optimization task (10). Only the candidate edges with top embedding distortions will be added into the latest graph. Since $z_{p,q}^{emb} = \|U_r^\top e_{p,q}\|_2^2 < \|U_N^\top e_{p,q}\|_2^2$, only the lower bound of embedding distortions can be estimated.

Step 4: Spectral stability checking. In this work, the edge embedding distortions are adopted for checking the spectral stability of the learned graph. If there exists no additional edge that has an embedding distortion greater than a given tolerance level (tol), GRASPEL iterations can be terminated. Note that choosing a smaller distortion tolerance will require more GRASPEL iterations (edges), which enables the embedding distances on the learned graph to more precisely encode the distances between the original data points.

5 EXPERIMENTS

In this section, extensive experiments have been conducted to evaluate the performance of the proposed GRASPEL algorithm for SC and DR applications. More details about the SC algorithm flow, data sets and the performance evaluation metrics used in our experiments can be found in the appendix sections.

5.1 Experiment Setup

Data preparation. Since this work primarily focuses on learning graphs from high-dimensional data points, the proposed method can be orthogonal to existing research related to deep learning based SC methods: an autoencoder can be first applied to transform the input data into more optimal features that can subsequently become the input of GRASPEL for learning graphs in SC. For fair comparisons with other state-of-the-art graph learning methods, we directly use the raw data as input without any additional pre-processing steps.

Input parameters. When applying Algorithm 1 to our data sets for the graph learning tasks shown in this section, we randomly sample candidate edges that connect between the top and bottom $0.05|V|$ ($\epsilon = 0.05$) nodes in the 1D sorted array according to the Fiedler vector, which allows GRASPEL to quickly identify the most spectrally-critical edges. Note that choosing a smaller ϵ value will allow more efficient edge sampling for estimating global graph (manifold) structural properties, while choosing a greater ϵ value will require more samples but may lead to better preservation of mid-to-short range graph (manifold) structural properties. When estimating the spectral distortion of each candidate edge we compute the first few ($2 \leq r \leq 10$) Laplacian eigenvectors for the spectral graph embedding step. We consider the edge sampling ratio $0.0 < \zeta < 1.0$ and add a few edges ($1 \leq s \leq 0.05|V|$) to the latest graph in each GRASPEL iteration. $\sigma = 10^3$ in (9) has been used for computing Θ in all experiments.

Results visualization. When creating the 3D spectral graph drawings (layouts), each entry of the first three nontrivial Laplacian eigenvectors (u_2, u_3, u_4) corresponds to the x, y and z coordinates of each node (data point), respectively. The ground-truth label of each data point is shown using its corresponding color. The edges

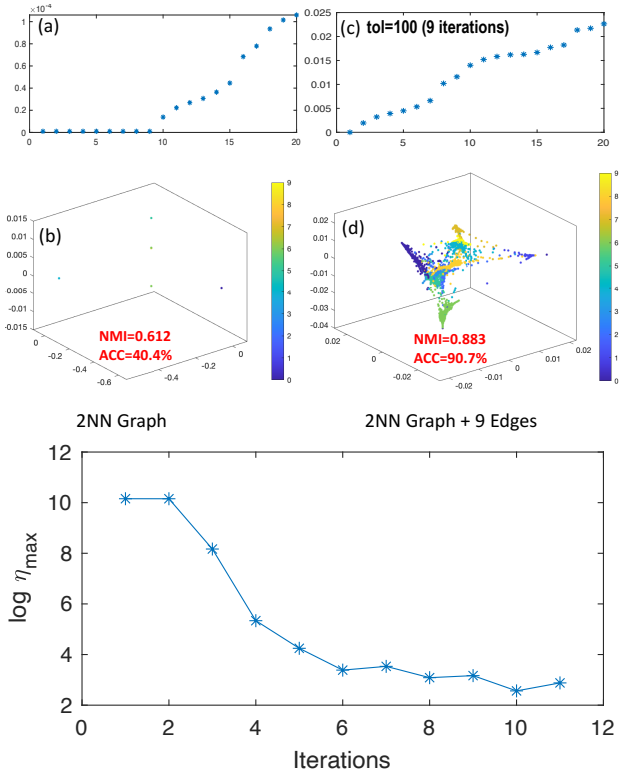

Figure 2: The first 20 Laplacian eigenvalues (top), spectral drawings (middle) and the embedding distortions (bottom) of the graph learned through GRASPEL iterations.

are omitted in the layouts to better reveal the structure of the data points (manifolds).

Computing platform. The following experiments are performed using MATLAB R2020b running on a Laptop with 10th Intel(R) Core(TM) i5 CPU and 8GB RAM.

5.2 Experiment Results

Spectral stability (convergence) checking. As shown in Figure 2, for the USPS data set, when starting from a 2NN initial graph GRASPEL identifies one additional edge to be included in each iteration and requires only 11 iterations (5 seconds) to effectively mitigate the maximum embedding distortion by over $1,000\times$. The rapidly improved NMI (Normalized Mutual Information) and ACC (clustering accuracy) in SC imply significantly improved graph quality [19]. Comparing with the state-of-the-art graph construction algorithm [10], our method is over $400\times$ faster while achieving much better solution in spectral (embedding) clustering tasks.

Spectral clustering results. Comprehensive results of SC using four graph learning (construction) methods are shown in Table 1. As observed, GRASPEL consistently achieves the state-of-the-art ACC/NMI results in SC. The graph density ($|E|/|V|$) results are also reported in Table 2, while the runtime reported in Table 3 includes the total time for eigendecomposition of the Laplacian matrix and k-means clustering. In Table 4, the runtime of the consensus method includes the time for consensus information calculation and edge

pruning, while the runtime of GRASPEL includes the total time for spectral graph densification. As observed, GRASPEL consistently achieves the state-of-the-art graph density and runtime results for SC tasks.

Table 1: Spectral Clustering (SC) Results

	ACC(%)/ NMI			
Data Set	Standard k-NN	Consensus [21]	LSGL [10]	GRASPEL
COIL20	75.72/0.86	81.60/0.90	85.49/**0.95**	**86.46**/0.94
PenDigits	74.36/0.79	71.08/**0.79**	74.53/0.77	**82.40**/**0.79**
USPS	64.31/0.79	68.54/0.81	81.50/0.84	**91.50**/**0.89**
MNIST	64.20/0.74	-	-	**74.63**/**0.78**

Table 2: Graph density results

	Graph density ($\|E\|/\|V\|$)			
Data Set	Standard k-NN	Consensus [21]	LSGL [10]	GRASPEL
COIL20	6.12	5.06	11.99	**1.39**
PenDigits	6.76	6.70	186.52	**2.96**
USPS	7.30	6.58	29.97	**1.70**
MNIST	7.46	-	-	**1.72**

Table 3: SC runtime results

	Spectral clustering (SC) time (seconds)			
Data Set	Standard k-NN	Consensus [21]	LSGL [10]	GRASPEL
COIL20	0.03	0.03	0.08	**0.02**
PenDigits	0.18	**0.16**	4.42	0.17
USPS	0.72	0.56	7.05	**0.28**
MNIST	252.59	-	-	**3.06**

Table 4: Graph learning (construction) time

	Graph construction time (seconds)		
Data Set	Consensus [21]	LSGL [10]	GRASPEL
COIL20	2.43	13.56	**0.29**
PenDigits	172.51	1,085.43	**2.04**
USPS	574.28	2,074.78	**3.37**
MNIST	-	-	**208.89**

Spectral embedding results. In Figure 3, we show the first few Laplacian eigenvalues and 3D spectral drawings of the graphs learned with different distortion tolerance levels for a subset (including all the 24, 462 handwritten digits from 0 to 6) of the MNIST data set and the test set of the Fashion MNIST data set (see Appendix A.2 for details) including 10, 000 article images from style 0 to 9 [32]. As observed, when starting from an initial 2NN graph, GRASPEL has dramatically mitigated the embedding distortions by adding only a few edges into the initial 2NN graph. By examining the first few Laplacian eigenvalues, we notice that the initial 2NN graph of the MNIST data set has nine connected components (that equals to the number of zero eigenvalues); with only 10 extra edges added via GRASPEL iterations, a well-connected graph can be formed for preserving the structure of the original data set. For the Fashion MNIST data set, the initial 2NN graph has six connected components, which becomes a well-connected graph with only seven edges added through GRASPEL iterations. It is also observed that when starting with initial 2NN graphs a few GRASPEL iterations have already dramatically improved clustering results: the NMI has been improved from 0.012 to 0.873 for the MNIST data set, 0.599 to 0.626 for the Fashion MNIST data set, and 0.04 to 0.840 for the Pendigit data set; the ACC has been improved from 16.1% to 90.7% for the MNIST data set, 43.9% to 55.5% for the Fashion MNIST data set, and 15.2% to 89.4% for the Pendigit data set.

Implications for dimensionality reduction. For the MNIST data set, we observe relatively large gaps between the 6th and 7th Laplacian eigenvalues, implying that the intrinsic dimensionality of the GRASPEL-learned graphs (manifolds) is approximately five, whereas for the Fashion MNIST data set the intrinsic dimensionality is approximately six. It is also observed that the GRASPEL iterations with different number (r) of Laplacian eigenvectors for spectral graph embedding using (9) and distortion computations always lead to similar gaps between eigenvalues. Consequently, we expect the GRASPEL-learned graphs to play an important role in revealing the intrinsic dimensionality of a data set and potentially lead to the development of highly-efficient tools for dimensionality reduction and data visualization.

Resistance-distance correlation. For the full USPS data set we evaluate graph learning quality by checking if the effective-resistance distances on the learned graph will properly encode the ℓ_2 distances between the original data points. To this end, we randomly pick up $1,000$ node pairs and compute their effective-resistance distances. To avoid choosing nearby nodes, we first sort nodes according to the Fiedler vector and then pick up the node pairs (n_{top}, n_{bot}) from the node sets N_{top} and N_{bot} formed with the top and bottom $5\%|V|$ sorted nodes, respectively. Then, these effective-resistance distances are compared with the corresponding ℓ_2 distances between the original data points by checking the Pearson correlation coefficient. In Figure 4, we observe that the effective-resistance distances on the graph learned via 100 GRASPEL iterations (27 seconds) exhibit the highest correlation ($R = 0.352$) with the ℓ_2 distance between the original data points, while the 3NN or 4NN graphs which are much denser only achieve $R = 0.063$ and $R = 0.093$, respectively. It is also observed that increasing k from 3 to 4 for constructing the kNN graph only improves the resistance correlation but not necessarily the clustering quality (e.g. NMI and ACC metrics), whereas the graph learned by GRASPEL iterations achieves the best results considering all aspects.

Multilevel t-SNE visualization. The t-Distributed Stochastic Neighbor Embedding (t-SNE) has become one of the most popular visualization tools for high-dimensional data analytic tasks [14, 16]. However, its high computational cost limits its applicability to large scale problems. To improve the scalability of t-SNE, a multilevel algorithm based on kNN graph coarsening has been introduced [33], which allows performing t-SNE for a much smaller set of representative data points corresponding to the nodes in the coarsened graphs. However, it would be quite challenging to determine the optimal k for constructing kNN graphs.

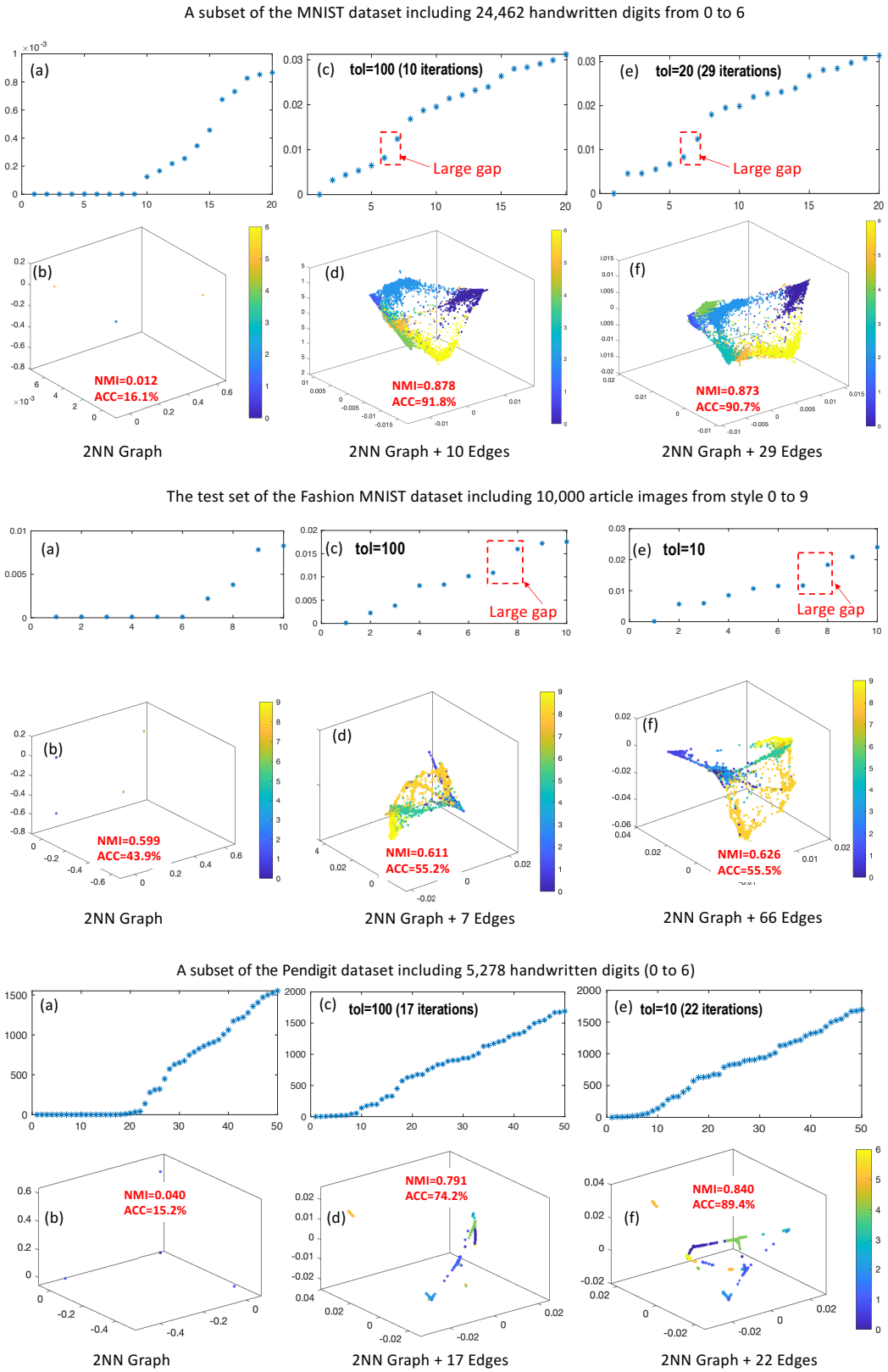

Figure 3: The first few eigenvalues (top) and spectral drawings (bottom) of the kNN and GRASPEL-learned graphs.

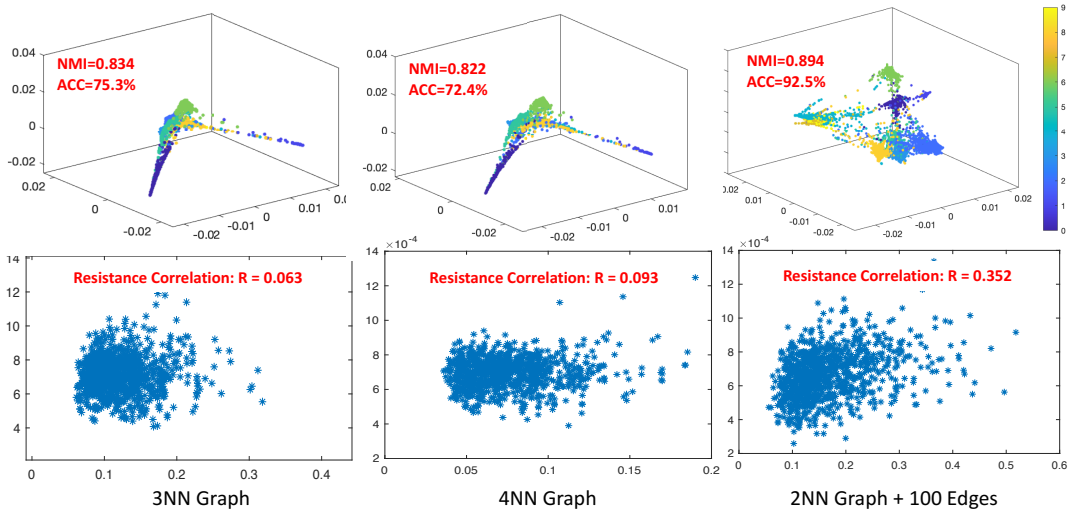

Figure 4: The Pearson correlation coefficients of resistance distances on the kNN and GRASPEL-learned graphs.

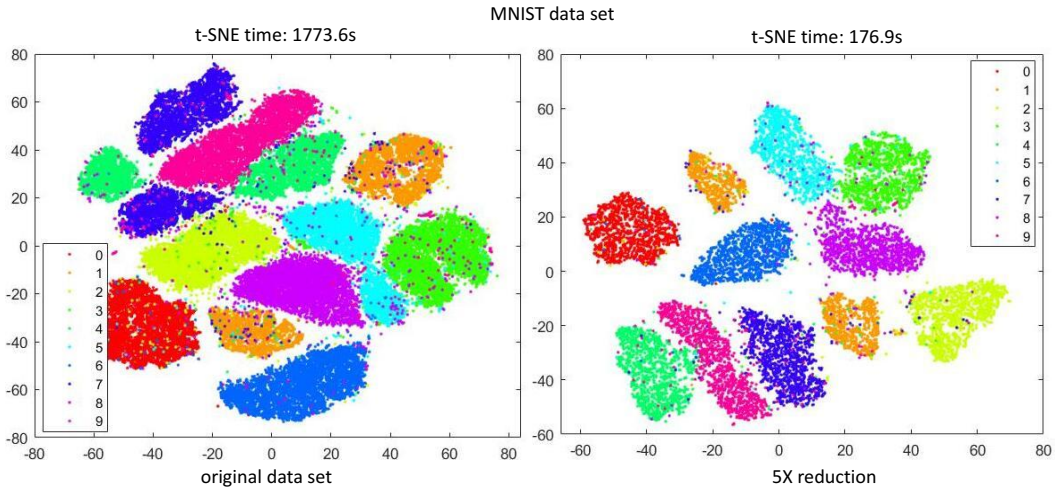

Figure 5: Multilevel t-SNE visualizations of the original (left) and reduced (right) MNIST data set.

In this work, we first learn a sparse graph (topology) with GRASPEL starting from an initial 2NN graph of the MNIST data set. Then a spectral graph coarsening procedure [33] is applied to produce hierarchical coarse-level graph representations. In the last, t-SNE visualization can be obtained by directly using the data points corresponding to the nodes of the coarsest graph. Figure 5 shows the visualization and runtime results of the standard t-SNE (with tree-based acceleration) [30] and the multilevel t-SNE algorithm. When using a $5X$ graph coarsening (reduction) ratio, we achieve 10.0× speedup for MNIST without loss of visualization quality.

6 CONCLUSION

In this work, we present a spectral approach (GRASPEL) for graph topology learning from potentially high-dimensional input data. We show that the graph topology learning problem can be solved through spectral graph densification by iteratively including the most spectrally-critical edges into the latest graph to mitigate the graph embedding distortion. Unlike traditional approaches which require at least $O(N^2)$ time in each iteration, each GRASPEL iteration allows identifying the most spectrally-critical edges in $O(N \log N)$ time. When comparing with state-of-the-art graph learning approaches, our approach shows more scalable runtime performance and always leads to substantially improved solution quality in SC and DR tasks.

ACKNOWLEDGMENT

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] C. Carey. *Graph Construction for Manifold Discovery*. PhD thesis, University of Massachusetts Amherst, 2017.

[3] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, and E. Y. Chang. Parallel spectral clustering in distributed systems. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):568–586, 2011.

[4] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst. Learning laplacian matrix in smooth graph signal representations. *IEEE Transactions on Signal Processing*, 64(23):6160–6173, 2016.

[5] 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.

[6] 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.

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

[8] T. Jebara, J. Wang, and S.-F. Chang. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 441–448. ACM, 2009.

[9] V. Kalofolias. How to learn a graph from smooth signals. In *Artificial Intelligence and Statistics*, pages 920–929, 2016.

[10] V. Kalofolias and N. Perraudin. Large scale graph learning from smooth signals. *International Conference on Learning Representations (ICLR 2019)*, 2019.

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

[12] S. Kumar, J. Ying, J. V. de Miranda Cardoso, and D. Palomar. Structured graph learning via laplacian spectral constraints. In *Advances in Neural Information Processing Systems*, pages 11651–11663, 2019.

[13] B. Lake and J. Tenenbaum. Discovering structure by learning sparse graphs. 2010.

[14] G. C. Linderman and S. Steinerberger. Clustering with t-sne, provably. *arXiv e-print, arXiv:1706.02582*, 2017.

[15] Y. Liu, Q. Gao, Z. Yang, and S. Wang. Learning with adaptive neighbors for image clustering. In *IJCAI*, pages 2483–2489, 2018.

[16] L. v. d. Maaten and G. Hinton. Visualizing Data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[17] M. Maier, U. V. Luxburg, and M. Hein. Influence of graph construction on graph-based clustering measures. In *Advances in neural information processing systems*, pages 1025–1032, 2009.

[18] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[19] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *NIPS*, 14(2):849–856, 2001.

[20] C. H. Papadimitrou and K. Steiglitz. Combinatorial optimization: algorithms and complexity. 1982.

[21] V. Premachandran and R. Kakarala. Consensus of k-nns for robust neighborhood selection on graph-based manifolds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1594–1601, 2013.

[22] M. G. Rabbat. Inferring sparse graphs from smooth signals with theoretical guarantees. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6533–6537. IEEE, 2017.

[23] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[24] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[25] 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.

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

[27] D. A. Spielman and S.-H. Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014.

[28] A. Strehl and J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[29] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[30] L. Van Der Maaten. Accelerating t-SNE Using Tree-based Algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[31] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[32] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[33] 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.

A APPENDIX

A.1 Spectral Clustering Algorithm Flow

Algorithm 2 Spectral Clustering Algorithm

Input: A graph $G = (V, E, w)$ and the number of clusters r.
Output: Clusters C_1,...,C_r.

1: Compute the adjacency matrix A, and diagonal matrix D;
2: Obtain the unnormalized Laplacian matrix L=D-A;
3: Compute the eigenvectors u_1,...,u_r that correspond to the bottom r nonzero eigenvalues of L;
4: Construct $U_r \in \mathbb{R}^{N\times r}$, with r eigenvectors of L stored as columns;
5: Perform k-means algorithm to partition the rows of U_r into r clusters and return the result.

A.2 Data Sets Description

COIL20: a data set contains 1, 440 gray-scale images of 20 objects, and each object on a turntable has 72 normalized gray-scale images taken from different degrees. The image size is 32x 32 pixels.
PenDigits: a data set consists of 7,494 images of handwritten digits from 44 writers, using the sampled coordination information. Each digit is represented by 16 attributes.
USPS: a data set includes 9, 298 scanned hand-written digits from 0 to 9 on the envelops from U.S. Postal Service with 256 attributes.
MNIST: a data set consists of 70,000 images of handwritten digits. Each image has 28-by-28 pixels in size. This database can be found at website (http://yann.lecun.com/exdb/mnist/).
Fashion-MNIST: is a dataset of Zalando's article images consisting of a training set of 60, 000 examples and a test set of 10, 000 examples [32]. Each example is a 28x28 grayscale image, associated with a label from 10 classes.

A.3 Algorithms for Comparison

Standard kNN: the most widely used affinity graph construction method. Each node is connected to its k nearest neighbors.
Consensus of kNN (cons-kNN) [21]: the state-of-the-art neighborhood selection methods to construct the affinity graphs. It selects strong neighborhoods to improve the robustness of the graph by using the consensus information from different neighborhoods in a given kNN graph.
LSGL [10]: a method to automatically select the parameters of the model introduced in [9] given a desired graph sparsity level. The default settings have been used in our experiments.

A.4 Evaluation Metric

The ACC metric measures the agreement between the clustering results generated by clustering algorithms and the ground-truth labels. The ACC can be computed by:

$$ACC = \frac{\sum_{i=1}^{n} \delta(y_i, map(c_i))}{n}, \tag{13}$$

where n is the number of samples in the data set, y_i is the ground-truth label provided by the data sets, and c_i is clustering result obtained from the algorithm. $\delta(x, y)$ is a delta function defined as: $\delta(x, y)$=1 for $x = y$, and $\delta(x, y)$=0, otherwise. $map(\bullet)$ is a permutation function that maps each cluster index c_i to a ground truth label, which can be realized using the Hungarian algorithm [20].
The NMI metric is in the range of [0, 1], while a higher NMI value indicates a better matching between the algorithm generated result and ground truth result. For two random variables P and Q, normalized mutual information is defined as [28]:

$$NMI = \frac{I(P, Q)}{\sqrt{H(P)H(Q)}}, \tag{14}$$

where $I(P, Q)$ denotes the mutual information between P and Q, while $H(P)$ and $H(Q)$ are entropies of P and Q.