Towards Scalable Spectral Embedding and Data Visualization via Spectral Coarsening

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

Ying Zhang
Stevens Institute of Technology
Hoboken, New Jersey
yzhan232@stevens.edu

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

ABSTRACT

This paper proposes a scalable multilevel framework for the spectral embedding of large undirected graphs. The proposed method first computes much smaller yet sparse graphs while preserving the key spectral (structural) properties of the original graph, by exploiting a nearly-linear time spectral graph coarsening approach. Then, the resultant spectrally-coarsened graphs are leveraged for the development of much faster algorithms for multilevel spectral graph embedding (clustering) as well as visualization of large data sets. We conducted extensive experiments using a variety of large graphs and datasets and obtained very promising results. For instance, we are able to coarsen the "coPapersCiteseer" graph with 0.43 million nodes and 16 million edges into a much smaller graph with only $13K$ ($32X$ fewer) nodes and $17K$ ($950X$ fewer) edges in about 16 seconds; the spectrally-coarsened graphs allow us to achieve up to $1,100X$ speedup for multilevel spectral graph embedding (clustering) and up to 60X speedup for t-SNE visualization of large data sets.

CCS CONCEPTS

• **Information systems** → **Clustering**; **Clustering and classification**; • **Human-centered computing** → **Visualization**; • **Mathematics of computing** → *Graph algorithms.*

KEYWORDS

Spectral graph theory; spectral embedding; graph clustering; data visualization

ACM Reference Format:
Zhiqiang Zhao, Ying Zhang, and Zhuo Feng. 2021. Towards Scalable Spectral Embedding and Data Visualization via Spectral Coarsening. In *Proceedings of the Fourteenth ACM International Conference on Web Search and Data Mining (WSDM '21), March 8–12, 2021, Virtual Event, Israel.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3437963.3441767

1 INTRODUCTION

Recent research shows that by leveraging the key spectral properties of eigenvalues and eigenvectors of graph Laplacians, more efficient algorithms can be developed for tackling many graph-related computing tasks [45]. For example, spectral methods can potentially lead to much faster algorithms for solving sparse matrices [40, 53], numerical optimization [7], data mining [32], graph analytics [22], machine learning [9], as well as very-large-scale integration (VLSI) computer-aided design (CAD) [13, 14, 51, 53]. To this end, spectral sparsification of graphs has been extensively studied in the past decade [3, 24, 41, 42] to allow computing almost-linear-sized [1] subgraphs or sparsifiers that can robustly preserve the spectrum. The sparsified graphs retain the same set of vertices but much fewer edges, which can be regarded as ultra-sparse graph proxies and have been leveraged for developing a series of nearly-linear-time numerical and graph algorithms [7, 15, 40, 41].

Another way of simplifying graphs is to directly reduce the size of the graphs, which is widely used in many areas, including graph partitioning [20], machine learning [9] and multigrid solvers [23, 28]. However, most of the graph coarsening techniques cannot guarantee the preservation of the spectral properties on the coarsened graphs, and much remains to be understood about the effect of the graph coarsening on the spectrum of a general graph.

In this paper, we introduce a scalable algorithmic framework *spectral coarsening of graphs* for dramatically reducing the size (both nodes and edges) of undirected graphs while preserving the key spectral (structural) properties of the original graph. The spectrally-coarsened graphs will immediately lead to the development of much faster numerical and graph-related algorithms. Based on the graph coarsening algorithm, multilevel frameworks for scalable spectral graph embedding and data visualization are proposed. The major contribution of this work has been summarized as follows:

- To well preserve the key spectral properties of the original graph in the coarsened graphs, a scalable framework for spectrum-preserving graph aggregation (coarsening) and sparsification is proposed for robustly constructing sparsified coarsened graphs that have much fewer number of nodes.
- Multilevel frameworks are proposed to allow leveraging spectrally-coarsened graphs for much faster spectral graph clustering as well as visualization of large data sets.
- We have obtained very promising experiment results for a variety of graph problems: the spectrally-coarsened graphs allow us to achieve up to **1,100X** speedup for spectral graph clustering and up to **60X** speedup for t-SNE visualization of large data sets.

2 RELATED WORK

There are two major ways to simplify a graph: graph sparsification aims to reduce the number of edges, while graph coarsening reduces the number of graph nodes. Graph sparsification and coarsening

[1] The number of vertices (nodes) is similar to the number of edges.

WSDM '21, March 8–12, 2021, Virtual Event, Israel

ACM ISBN 978-1-4503-8297-7/21/03…$15.00
https://doi.org/10.1145/3437963.3441767

Table 1: Symbols and their denotations in this work

Symbol	Denotation	Symbol	Denotation
$G = (V, E_G, w_G)$	Original graph	$\mathbf{L_G}$	Lap. of G
$P = (V, E_P, w_P)$	Spectrally-spar. G	$\mathbf{L_P}$	Lap. of P
$R = (V_R, E_R, w_R)$	Coarsened G w/o spar.	$\mathbf{L_R}$	Lap. of R
$S = (V_R, E_S, w_S)$	Coarsened G w/ spar.	$\mathbf{L_S}$	Lap. of S
$\mathbf{H_G^R} \in \mathbb{R}^{\lvert V_R\rvert \times \lvert V\rvert}$	G-to-R mapping	$\mathbf{H_R^G} \in \mathbb{R}^{\lvert V\rvert \times \lvert V_R\rvert}$	R-to-G mapping

have been widely used in the applications of graph clustering and partitioning [10, 20, 36, 50], as well as data (graph) visualization [16, 21, 49].

Different graph sparsification techniques have been proposed. Graph spanners [12, 31] were proposed to preserve the pair distances between nodes. Benczur and Karger [4, 5] then introduced the cut sparsifier, which can preserve cut values between the original graph and the sparsified graph. Later, Spielman and Teng [42] proposed the spectral sparsifier for preserving the key eigenvalues and eigenvectors, which is a stronger notation than the cut sparsifier. Since then, more spectral related sparsification methods are proposed, like the spectral preservation of pseudoinverse for the graph Laplacian by Li [25].

Compared to the solid theoretical work on the graph sparsification, graph coarsening is harder to understand due to the lack of matured theoretical frameworks. A variety of spectral coarsening schemes have been proposed, but the majority of the algorithms are based on heuristics. [11] proposed the Kron reduction of the graph based on the Schur complement. Purohit et al. [34] introduced the CoarseNet that is able to coarsen graphs while preserving the largest eigenvalue of its adjacency matrix, such that the diffusion characteristics of the original graph can be kept. Loukas and Vandergheynst [29] proposed a theoretical framework which proves the spectral preservation of the original graph after coarsening based on the concept of the restricted spectral similarity. Recently, Bravo-Hermsdorff and Gunderson [17] proposed a unified framework of graph sparsification and coarsening, which aims to preserve the Laplacian pseudoinverse on the coarsened graph.

3 BACKGROUND: UNDIRECTED GRAPH

Given an undirected graph $G = (V, E_G, w_G)$ with V denoting the set of vertices, E_G denoting the set of undirected edges, and w_G denoting the associated edge weights, we define $\mathbf{D_G}$ to be a diagonal matrix such that $D_G(i,i)$ equals to the (weighted) degree of node i, while $\mathbf{A_G}$ and $\mathbf{L_G}$ denote the adjacency and Laplacian matrices of undirected graph G , respectively:

$$\mathbf{A_G}(i,j) = \begin{cases} w_G(i,j) & \text{if } (i,j) \in E_G \\ 0 & \text{otherwise .} \end{cases} \tag{1}$$

Graph Laplacians can be constructed by using $\mathbf{L_G} = \mathbf{D_G} - \mathbf{A_G}$.

Spectral sparsification aims to find a spectrally-similar subgraph (sparsifier) $P = (V, E_P, w_P)$ that has the same set of vertices of the original graph $G = (V, E_G, w_G)$, but much fewer edges. We say G and its subgraph P are $\sigma-$spectrally similar if the following condition holds for all real vectors $\mathbf{x_G} \in \mathbb{R}^{|V|}$:

$$\frac{\mathbf{x_G}^\top \mathbf{L}_P \mathbf{x_G}}{\sigma} \leq \mathbf{x_G}^\top \mathbf{L}_G \mathbf{x_G} \leq \sigma \mathbf{x_G}^\top \mathbf{L}_P \mathbf{x_G}, \tag{2}$$

where $\mathbf{L}_G$ and $\mathbf{L}_P$ denote the Laplacian matrices of graph G and P, respectively. For better understanding, all symbols used in the paper are shown in Table 1. Define the relative condition number of $\mathbf{L}_G$ and $\mathbf{L}_P$ as $\kappa(\mathbf{L}_G, \mathbf{L}_P) = \lambda_{\max}/\lambda_{\min}$, where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest nonzero eigenvalues of $\mathbf{L}_G\mathbf{u} = \lambda\mathbf{L}_P\mathbf{u}$, and $\mathbf{u}$ is the generalized eigenvector of $\mathbf{L_G}$. It can be further shown that $\kappa(\mathbf{L}_G, \mathbf{L}_P) \leq \sigma^2$, which indicates that a smaller relative condition number or σ^2 corresponds to a higher spectral similarity.

Graph coarsening aims to find a smaller graph $R = (V_R, E_R, w_R)$ to approximate a larger graph $G = (V, E_G, w_G)$ through the graph mapping operator $\mathbf{H_G^R}$:

$$\mathbf{L_R} = \mathbf{H_G^R} \mathbf{L_G} (\mathbf{H_G^R})^\top, \tag{3}$$

where $\mathbf{H_G^R}$ is a coarsening matrix containing only 0 and 1. Also, coarsening process is a surjective mapping of the node set, where $(\mathbf{H_G^R})_{p,q} = 1$ if node q in graph G is aggregated to super-node p in graph R, and $(\mathbf{H_G^R})_{p',q} = 0$ for all nodes $p' \in \{v \in R : v \neq p\}$. Coarsened graph R and graph G satisfy restricted spectral similarity shown as the following condition [29]:

$$\frac{\mathbf{x_G}^\top \mathbf{L}_G \mathbf{x_G}}{\sigma'} \leq \mathbf{x_R}^\top \mathbf{L}_R \mathbf{x_R} \leq \sigma' \mathbf{x_G}^\top \mathbf{L}_G \mathbf{x_G}, \quad \forall \mathbf{x_R} \in \mathbf{U^R} \ \ \forall \mathbf{x_G} \in \mathbf{U^G}, \tag{4}$$

where $\mathbf{U^R} = \left[\mathbf{u_R^{(1)}}, \mathbf{u_R^{(2)}}, ..., \mathbf{u_R^{(k)}}\right]$ and $\mathbf{U^G} = \left[\mathbf{u_G^{(1)}}, \mathbf{u_G^{(2)}}, ..., \mathbf{u_G^{(k)}}\right]$ include the first k eigenvectors of $\mathbf{L_R}$ and $\mathbf{L_G}$ correspondingly.

4 SPECTRAL GRAPH COARSENING

4.1 Overview of our approach

This work introduces a *spectral graph coarsening* framework that allows computing much smaller yet spectrally-similar graph S. The Laplacian matrices of the corresponding graphs have been shown in Table 1 that also includes the graph mapping matrices. Our approach for spectral coarsening of undirected graphs includes the following three steps:

- **Step A** will determine the fine-to-coarse graph mapping operator and coarsen the original graph into a much smaller graph.
- **Step B** will extract spectrally-similar sparsifiers of the original (coarsened) graph and scale up the edge weights in the sparsified graphs.
- **Step C** will globally scale up edge weight of the coarsened graph.

Since the spectral node aggregation metric cannot be directly applied to relatively dense graphs [6], our approach will first examine the average node degrees in the original graph: if the original graph is relatively sparse ($|E_G| < 40|V|$), **steps A to B** will be performed in sequence; otherwise, if the original graph is too dense ($|E_G| > 40|V|$), **step B** will be performed first, which is followed by **step A**. Finally, the coarsened graph will be scaled up by **step C**.

4.2 Step A: spectrum-preserving aggregation

In this step, a multilevel spectral graph coarsening process will be performed until the desired size of the coarsened graph is reached. The graph mapping operators on each level $(\mathbf{H}_1^2, \cdots, \mathbf{H}_{m-1}^m)$ can

be created and leveraged for constructing a series of spectrally-coarsened graphs $G_1, G_2, \cdots, G_m(R)$, where G_1 is the original graph, and $|V_1| = N > |V_2| > \cdots > |V_m|$. Notice that mapping operator $\mathbf{H}_i^{i+1} \in \{0,1\}^{|V_{i+1}| \times |V_i|}$ is a coarsening matrix containing only 0 and 1. It has following properties:

- Row (column) index of $\mathbf{H}_i^{i+1}$ corresponds to the node index in graph G_{i+1} (G_i).
- It is a surjective mapping of the node set, where $(\mathbf{H}_i^{i+1})_{p,q} = 1$ if node q in graph G_i is aggregated to super-node p in graph G_{i+1}, and $(\mathbf{H}_i^{i+1})_{p',q} = 0$ for all nodes $p' \in \{v \in V_{i+1} : v \neq p\}$.
- It is a locality preserving operator, where the subgraph of G_i induced by the non-zero entries of $(\mathbf{H}_i^{i+1})_{p,:}$ is connected for each $p \in V_{i+1}$.

For example, the coarser graph Laplacian $\mathbf{L}_{G_{i+1}}$ can be computed by

$$\mathbf{L}_{G_{i+1}} = \mathbf{H}_{\mathbf{i}}^{\mathbf{i+1}} \mathbf{L}_{G_i} \mathbf{H}_{\mathbf{i+1}}^{\mathbf{i}}, \quad \mathbf{H}_{\mathbf{i+1}}^{\mathbf{i}} = (\mathbf{H}_{\mathbf{i}}^{\mathbf{i+1}})^{\mathbf{T}}. \tag{5}$$

Graph coarsening via local spectral embedding. Since $\mathbf{H}_i^{i+1}$ is a locality-preserving operator, how to construct the $\mathbf{H}_i^{i+1}$ is the key problem. In this work, we leverage an efficient yet effective local spectral embedding scheme to identify node clusters based on emerging graph signal processing techniques [39].

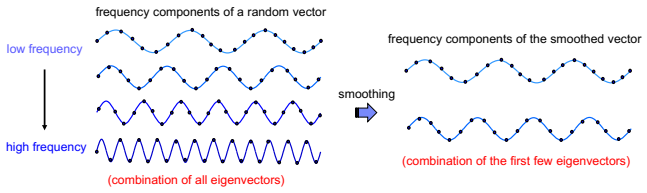

Figure 1: Smoothing a random vector on a path graph.

As shown in Figure 1, we consider a random vector (graph signal) $\mathbf{x}$ which can be expressed with a linear combination of eigenvectors $\mathbf{u_i}$, for $i = 1, ..., N$, of a path-graph Laplacian. Low-pass graph filters can be adopted to quickly filter out the "high-frequency" components of the random graph signal or the eigenvectors corresponding to high eigenvalues of the graph Laplacian. To this end, rather simple smoothing functions, such as the Gauss-Seidel and Jacobi methods, can be used. By applying the smoothing function on $\mathbf{x}$, a smoothed vector $\tilde{\mathbf{x}}$ can be obtained in linear time, which can be considered as a linear combination of the first few eigenvectors:

$$\mathbf{x} = \sum_{i=1}^{N} \beta_i \mathbf{u_i} \quad \xrightarrow{\text{filtering}} \quad \tilde{\mathbf{x}} = \sum_{i=1}^{n} \tilde{\beta}_i \mathbf{u_i} \qquad n \leq N. \tag{6}$$

More specifically, given a set of k initial random vectors $\mathbf{X} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k)})$ that are orthogonal to the all-one vector, we can obtain the smoothed vectors in $\tilde{\mathbf{X}}$ by applying a few steps of Gauss-Seidel iterations for solving the linear system of equations $\mathbf{L}_G \mathbf{x}^{(i)} = \mathbf{0}$. Based on the smoothed vectors in $\tilde{\mathbf{X}}$, each node is embedded into a k-dimensional space such that nodes p and q are considered spectrally similar if their low-dimensional embedding vectors $\tilde{\mathbf{X}}_{\mathbf{p},:} \in \mathbb{R}^k$ (the p-th row of $\tilde{\mathbf{X}}$) and $\tilde{\mathbf{X}}_{\mathbf{q},:} \in \mathbb{R}^k$ (the q-th row of $\tilde{\mathbf{X}}$) are highly correlated. Consequently, spectrally-similar nodes p and q can be then aggregated together for node reduction purpose. Here the node distance is measured by the spectral node affinity $d_{p,q}$ for neighboring nodes p and q [6, 28]:

$$d_{p,q} = \frac{(\tilde{\mathbf{X}}_{\mathbf{p},:}, \tilde{\mathbf{X}}_{\mathbf{q},:})^2}{(\tilde{\mathbf{X}}_{\mathbf{p},:}, \tilde{\mathbf{X}}_{\mathbf{p},:})(\tilde{\mathbf{X}}_{\mathbf{q},:}, \tilde{\mathbf{X}}_{\mathbf{q},:})} \tag{7}$$

where $(\tilde{\mathbf{X}}_{\mathbf{p},:}, \tilde{\mathbf{X}}_{\mathbf{q},:})$ is the inner product.

4.3 Step B & C: spectral sparsification & scaling

The proposed node aggregation scheme in Section 4.2 will enable us to reliably construct smaller graphs that have fewer vertices. However, the aggregated nodes may potentially result in much denser graphs (with significantly higher node degrees), which may incur even greater computational and memory cost for graph operations.

To address the challenges from relatively dense graphs, we propose the following highly effective yet scalable algorithms in **step B**: the nearly-linear time spectral graph sparsification and subgraph scaling schemes for handling dense graphs G. Note that when **step B** is applied for a sparse input graph, the same procedures can be applied to the coarsened graph R (with potentially higher density) for computing S after the node aggregation scheme or the fine-to-coarse graph mapping operators are determined.

It has been shown that every graph has a low-stretch spanning tree (LSST) that can be leveraged as an initial sparsifier with a bounded total stretch [43]:

$$\kappa(\mathbf{L}_G, \mathbf{L}_P) \leq \mathrm{Tr}(\mathbf{L}_P^{+}\mathbf{L}_G) = \mathrm{st}_P(G) \leq (m \log n \log\log n), \tag{8}$$

where $m = |E_G|$, $n = |V|$, and $\mathrm{Tr}(\mathbf{L}_P^{+}\mathbf{L}_G)$ is the trace of $\mathbf{L}_P^{+}\mathbf{L}_G$. Such a result motivates the construction of an ultra-sparse yet spectrally-similar subgraphs by recovering only a small portion of important off-tree edges to the spanning tree, which can dramatically reduce the mismatch between the original graph and the sparsifier [13, 14].

To further improve the quality of the coarsened graph with the minimum number of edges, an iterative edge weight scaling scheme [54] using constrained Stochastic Gradient Descent (SGD) with momentum as well as a global post-scaling process [52] can be applied for better matching the spectral properties of the original graph, leading to the improved approximation of the first few Laplacian eigenvalues and eigenvectors within the coarsened graph.

4.4 Algorithm complexity

The algorithm complexity of **step A** for the spectrum-preserving node aggregation procedure is $O(|E_P|)$ for dense graphs and $O(|E_G|)$ for sparse graphs, while the complexity of **step B** for spectral graph sparsification and edge scaling by SGD iterations is $O(|E_G| \log(|V|))$ for dense graphs and $O(|E_S| \log(|V_R|))$ for sparse graphs. The complexity of edge post-scaling is $O(|E_G|)$ for **step C** by leveraging the latest graph-theoretic Laplacian solvers [28, 53]. Therefore, the worse-case algorithm complexity of the proposed spectral graph coarsening method is $O(|E_G| \log(|V|))$.

5 MULTILEVEL SPECTRAL GRAPH EMBEDDING AND VISUALIZATION

In this section, multilevel frameworks that leverage spectrally-coarsened graphs for scalable spectral graph embedding (clustering) as well as data visualization of large data sets are introduced.

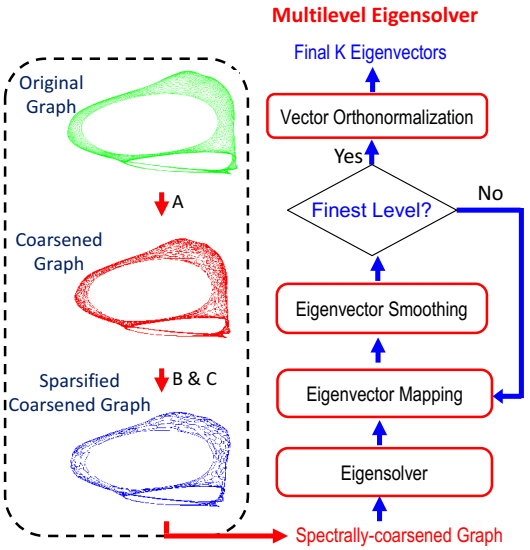

Figure 2: Multilevel graph Laplacian eigensolver.

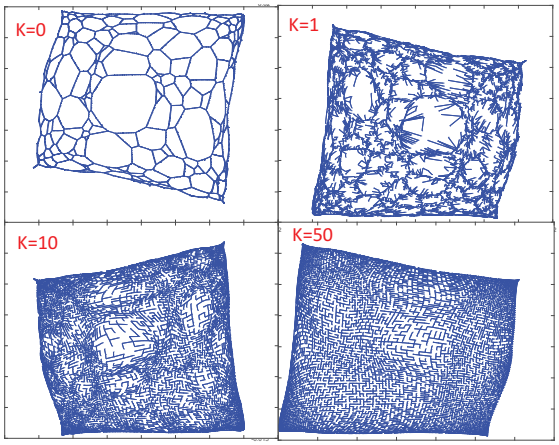

Figure 3: Solution refinement for spectral graph embedding with the first two Laplacian eigenvectors.

5.1 Scalable graph Laplacian eigensolver

We proposed a multilevel Laplacian eigensolver to calculate the first few nontrivial eigenvectors of the original graph Laplacian, shown in Figure 2. Instead of directly computing the first k eigenvectors of the original graph, we will first coarsen the original graph G into a much smaller graph S such that the eigenvectors of the coarsened graph can be efficiently calculated. Next, we will map the eigenvectors of the coarsened graph Laplacian onto a finer level using the graph mapping operators. To further improve the approximation quality of these eigenvectors, an eigenvector refinement (smoothing) procedure is applied. In this work, we adopt a weighted Jacobi iteration scheme for filtering out the high-frequency error signals on graphs [35]. The eigenvector mapping and smoothing procedures are recursively applied until the finest-level graph is reached. Finally, all eigenvectors computed for the finest-level graph will be orthonormalized using the Gram-Schmidt process.

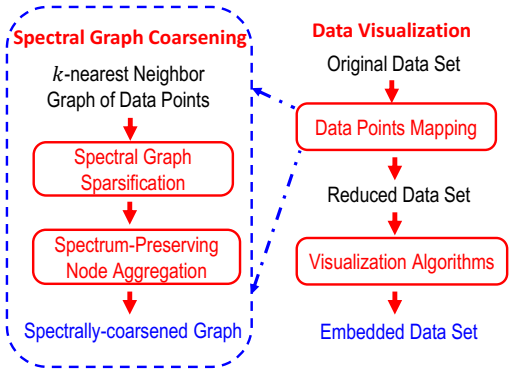

Figure 4: Multilevel visualization algorithm.

Figure 3 shows the 2D spectral embedding [2] results of a (sparsified) 2D mesh graph using the proposed eigenvectors solution refinement scheme. K-step weighted Jacobi relaxations have been applied to improve the eigenvector accuracy with K=0, 1, 10, and 50, respectively. Such results indicate that the approximate eigenvectors obtained from sparsified (coarsened) graphs can be significantly improved via the proposed solution refinement procedure.

Algorithm 1 Multilevel Laplacian Eigensolver

Input: $\mathbf{L}_{G_1}, \cdots, \mathbf{L}_{G_m}$, $\mathbf{H}_2^1, \cdots, \mathbf{H}_m^{m-1}$, k;

Initialize: $j := m$, $\mathbf{B}_{G_v} := \mathbf{I}$ for ratio cut [48] or $\mathbf{B}_{G_v} := \mathbf{D}_{G_v}$ for normalized cut [48], where $v = 1, \cdots, m$;
Compute the first k eigenpairs $(\lambda_1^m, \mathbf{u}_1^m), \cdots, (\lambda_k^m, \mathbf{u}_k^m)$ of the eigenvalue problem $\mathbf{L}_{G_m}\mathbf{u}_i^m = \lambda_i^m \mathbf{B}_{G_m}\mathbf{u}_i^m$ for $i = 1, \cdots k$;
Form matrix $\mathbf{U}^m$ with the first k vectors $\mathbf{u}_1^m, \cdots, \mathbf{u}_k^m$ as its columns;
while $j > 1$ **do**
  Map $\mathbf{U}^j$ from level j to level $j-1$ by $\mathbf{U}^{j-1} = \mathbf{H}_j^{j-1}\mathbf{U}^j$;
  for $i = 1$ **to** k **do**
    $\mathbf{y} := \mathbf{U}^{j-1}[:, i]$, which is the i-th column of $\mathbf{U}^{j-1}$;
    Filter vector $\mathbf{y}$ by performing a few weighted-Jacobi iterations to $(\mathbf{L}_{G_{j-1}} - \lambda_i^m \mathbf{B}_{G_{j-1}})\mathbf{y} = 0$;
    Update $\mathbf{U}^{j-1}[:, i]$ with the smoothed vector $\mathbf{y}$;
  end for
  $j := j - 1$;
end while
Perform orthonormalization to columns of $\mathbf{U}^1$;
Return $\mathbf{U} = \mathbf{U}^1$.

The detailed algorithm for multilevel Laplacian eigensolver is shown in Algorithm 1. The inputs of the algorithm include the Laplacian matrix of each hierarchical level $\mathbf{L}_{G_v} = \mathbf{D}_{G_v} - \mathbf{A}_{G_v}$, where $v = 1, \cdots, m$; mapping operator $\mathbf{H}_v^{v-1}$ from level v to level $v-1$; and the number of eigenvectors k. In the last, spectral graph clustering can be performed using the eigenvectors computed by Algorithm 1 in the subsequent k-means clustering step.

5.2 Multilevel algorithm for data visualization

Visualization of high-dimensional data is a fundamental problem in data analysis and has been used in many applications. For example, the t-Distributed Stochastic Neighbor Embedding (t-SNE)

[2] Drawing graphs with the first two nontrivial eigenvectors as the X- and Y-coordinates for each node [22].

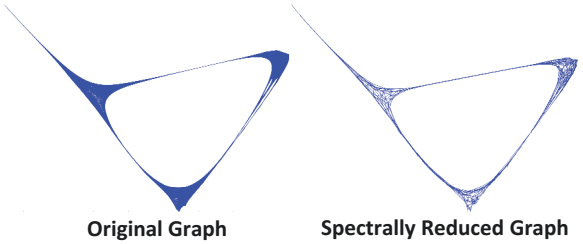

Figure 5: Spectral drawings of the original and coarsened "fe_ocean" (24X node reduction and 58X edge reduction).

[47] and LargeVis [44] have become the most effective visualization tools due to their capability of performing dimensionality reduction. However, these algorithms may suffer from very high computational cost for visualizing large real-world data sets due to the high computational complexity [30, 47].

Recent research [27] shows that the low-dimensional data points embedding obtained with t-SNE is closely related to the first few eigenvectors of the corresponding graph Laplacian that encodes the manifold of the original high-dimensional data points. This motivates us to propose a multilevel visualization algorithm based on our graph coarsening method, as shown in Figure 4. The idea is that data points closely related to each other on the manifold will be aggregated into much smaller sets, such that visualizing the reduced data set using existing tools such as t-SNE will be much faster and produce similar embedding results. To this end, we start by constructing a nearest-neighbor (NN) graph, such as the k-NN graph, for the original high-dimensional data points; then, a spectrally-coarsened (NN) graph is computed using the proposed spectral coarsening algorithm. Note that for k-NN graphs, the graph sparsification and scaling procedure (step B) will be performed before the spectral node aggregation step (step A). The detailed algorithm is shown in Algorithm 2.

Algorithm 2 Multilevel Data Visualization

Input: Original data set $\mathbf{F}$, number of neighbors k ;

1: Generate k-nearest neighbor (k-NN) graph G based on the data set F;
2: Generate the spectrally-coarsened graph S;
3: Obtain the corresponding mapping operator $\mathbf{H}_G^R$;
4: Form a reduced data set $\mathbf{F}_R$ by $\mathbf{F}_R = \mathbf{H}_G^R\mathbf{F}$;
5: Embed data points with any existing visualization tools on the reduced data set $\mathbf{F}_R$;
6: Return embedded data points for visualization.

6 EXPERIMENTAL RESULTS

In this section, extensive experiments have been conducted to evaluate the proposed spectral graph coarsening and spectral clustering methods with various types of graphs from the DIMACS10 graph collection[1, 2]. Graphs are from different applications, such as finite-element analysis problems ("fe_rotor") [8], numerical simulation graphs ("auto"), clustering graphs ("uk") and social network graphs ("coAuthorsDBLP" and "coPapersCiterseer") [8], etc. All experiments have been conducted on a single CPU core of a computing platform running 64-bit RHEW 6.0 with 2.67GHz 12-core CPU and 48GB DRAM memory.

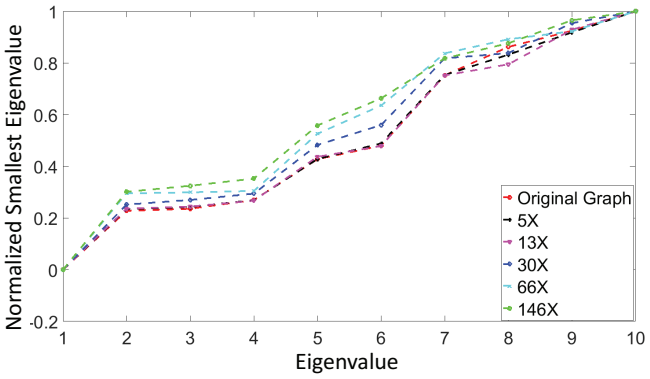

Figure 6: The first 10 normalized eigenvalues of the "fe_rotor" graph under different node reduction ratios.

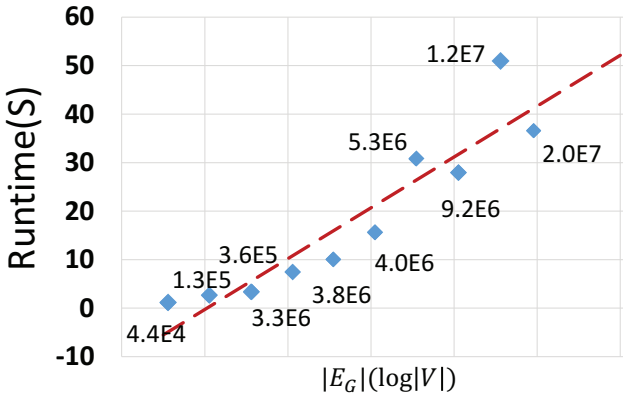

Figure 7: Scalability of the proposed coarsening method.

6.1 Results of spectral graph coarsening

Figure 5 shows the spectral drawings [22] of the "fe_ocean" graph and its coarsened graph computed by the proposed coarsening algorithm, where the node and edge reduction ratio are 24X and 58X, respectively. We observe that the spectral drawings of two graphs are highly similar to each other, which indicates very well preserved spectral properties (Laplacian eigenvectors) in the coarsened graph.

Figure 6 shows the first few normalized eigenvalues of the original and coarsened graph Laplacians, indicating clearly that the smallest eigenvalues of the original Laplacian and the coarsened Laplacians match very well even for very large reduction ratios.

Table 2 shows spectral graph coarsening results on different kinds of graphs using the proposed method, where T_r denotes the graph coarsening time. Compared to other test cases that correspond to sparse graphs, the graph density of "$appu^*$" is much higher and thus has been processed as a dense graph. We want to further emphasize that directly applying the prior node aggregation scheme will not produce acceptable results. For example, the node aggregation algorithm failed to generate the coarsened graph for "$appu^*$" due to very high graph density. On the other hand, there will be no issue for dense graphs if we apply **step B** for spectral graph sparsification and scaling before the node aggregation step.

Figure 7 shows the total spectral graph coarsening time with different problem sizes ($|E_G|\log(|V|)$) for various graphs, where

$|E_G|$ ($|V|$) denotes the number of edges (nodes) of the original graphs, respectively. As observed, the total spectral coarsening runtime increases almost linearly with the problem size, indicating the highly scalable performance of the proposed method.

6.2 Spectrum approximation

We also compared the performance of our proposed method with the following state-of-the-art graph coarsening methods: **(1)** Local variation based graph coarsening method [29]. Based on the concept of restricted spectral approximation, two possible graph contraction methods were proposed: edge-based contraction and neighborhood-based contraction. **(2)** Heavy edge matching based graph coarsening method, which is widely used for graph partitioning [19] and more recently in graph embedding [26]. **(3)** Kron reduction method [38]. The benefit of this method is that it can preserve the important spectral properties; however, the densities of coarsened graphs will be dramatically increased.

To measure the performance of different spectral coarsening methods, the mean relative eigenvalue errors between original graphs and coarsened graphs are reported in Table 3, where five methods are tested, including local variation with edge and neighborhood contraction, heavy edge contraction, Kron reduction, as well as our proposed coarsening method; r represents the node coarsening ratio, which can be calculated by $1 - |V_S|/|V|$; $|V|$ and $|V_S|$ are the number of node for the original graph and the coarsened graph, respectively. Given the first k eigenvalues ω and $\tilde{\omega}$ of the original graph and the coarsened graph, the mean relative error can be calculated by $\frac{1}{k}\sum_{i=1}^{k}\frac{|\omega_i-\tilde{\omega}_i|}{\omega_i}$ [29]. Four different graphs including airfoil ($|V| = 4,000$, $|E_G| = 11,490$) [33], yeast ($|V| = 1,458$, $|E_G| = 1,948$) [18], bunny ($|V| = 2,503$, $|E_G| = 65,490$) [46] and Minnesota ($|V| = 2,642$, $|E_G| = 3,304$) are tested in the experiment. We can observe that the spectrum can be better preserved on the coarsened graphs using our proposed graph coarsening algorithm compared to other methods. Table 4 shows the number of the edges for the coarsened graphs when using the different coarsening methods. We can observe that our method can achieve better graph sparsity when comparing to other methods.

6.3 Results of spectral embedding (clustering)

We evaluated the performance of the proposed spectral graph clustering algorithm on a variety of graphs from the DIMACS10 graph collection. We choose to partition all the graphs into 30 clusters. The built-in `eigs` and `kmeans` MATLAB functions are used for solving the eigenvalue problem and node clustering tasks, respectively. The normalized cut [37] is used to measure the quality of clusters, where a smaller value of normalized cut represents better clustering quality. Three methods have been tested, including spectral clustering with original graphs (no reduction), spectral clustering with spectrally-coarsened graphs generated by the proposed spectral coarsening technique, as well as the spectral clustering with coarsened graphs generated by METIS [19] with default settings. Note that we choose to coarsen the graphs with similar node reduction ratios when applying two coarsening frameworks, even though the spectrally-coarsened graphs have much fewer edges. Once the coarsened graphs are generated by two frameworks, the multilevel eigensolver will be leveraged for further spectral clustering.

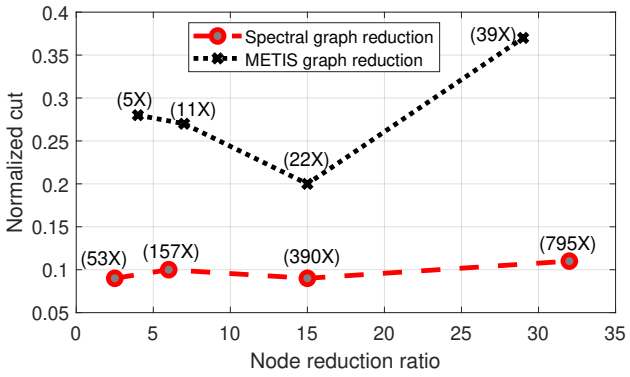

Figure 8: Clustering qualities (normalized cut) under different reduction ratios for the "coPapersCiteseer" graph [8], where $(\cdot)$ shows the edge reduction ratio.

Detailed experimental results are shown in Table 2. The performance of clustering is evaluated based on the normalized cut and its execution time. In the table, θ is the normalized cut; T_{eigs} are the execution time for solving the eigenvalue problems given the original graphs or coarsened graphs; T are the total execution time for spectral graph clustering including solving eigenvalue problems and clustering with k-means; "NA" denotes the failure of solving eigenvalue problems due to the limited memory resources. From the table, we can observe that spectrally-coarsened graphs can achieve consistently better clustering quality than the coarsened graphs do generated by METIS, indicating that our method can achieve better spectrum preservation with much fewer edges. Meanwhile, the superior sparsity of the spectrally-coarsened graphs enables better efficiency. The overall quality of generated clusters using the original graphs and the spectrally-coarsened graphs is similar to each other, but the cost when using the coarsened graphs is much lower than using the original graphs, especially for large graphs. For example, we achieve over $1,100X$ runtime speedup on the "smallworld" graph clustering. For larger graphs, such as the "coPapersCiteseer" graphs, spectral clustering without coarsening will fail due to the extremely high computation (memory) cost.

From the table, we can also conclude that most of the runtime is due to the eigensolver if the original graph is used, while the k-means and smoothing time will be dominant when using the spectrally-coarsened graph. However, the smoothing procedure is inherently highly parallel, making it possible to further improve the efficiency of the proposed spectral clustering and to develop high-quality parallel spectral clustering algorithms.

We evaluated the performance of the proposed spectral clustering method with different coarsening ratios, as shown in Figure 8 and Figure 9. In Figure 8, the normalized cut is presented with different size of coarsened graphs generated by the proposed method and METIS. The edge reduction ratios are also included in the figure. Figure 9 shows the runtime of the proposed method with the corresponding coarsened graphs. We observe that the proposed method can constantly produce better coarsened graphs with superior sparsity than METIS, which eventually leads to the better clustering results. Also, as shown in Figure 9, higher reduction ratios result in lower cost for graph coarsening as well as spectral clustering, while still maintaining high clustering quality. This indicates a very

Table 2: Spectral graph coarsening and clustering results with the best results highlighted in red and blue colors.

Graph	Original Graph (G)					Spectrally-coarsened Graph (S)						Coarsened Graph (M) by METIS					
	$\|V\|$	$\|E_G\|$	θ	T_{eigs}	T	$\frac{\|V\|}{\|V_S\|}$	$\frac{\|E_G\|}{\|E_S\|}$	T_r	θ	T_{eigs}	T	$\frac{\|V\|}{\|V_M\|}$	$\frac{\|E_G\|}{\|E_M\|}$	T_r	θ	T_{eigs}	T
fe_rotor	1.0E5	6.6E5	1.51	20.2s	22.8s	71X	180X	1.3s	**1.50**	0.2s	**2.9s**	51X	43X	1.6s	1.67	0.2s	9.4s
auto	4.5E5	3.3E6	1.10	479.7s	495.8s	30X	167X	14.8s	**1.08**	0.6s	**29.0s**	27X	24X	7.5s	1.60	3.5s	53.4s
uk	4.8E3	6.8E3	**1.01**	0.2s	0.6s	40X	51X	0.2s	1.03	0.1s	0.6s	34X	24X	0.1s	1.20	0.1s	**0.3s**
vsp_barth5	3.2E4	1.0E5	3.12	14.4s	16.6s	57X	122X	0.5s	**2.72**	0.2s	2.7s	44X	9X	0.3s	2.94	0.3s	**2.4s**
smallworld	1.0E5	5.0E5	**6.92**	1.6E4s	1.6E4s	22X	5X	32.2s	6.93	9.2s	**11.4s**	28X	4X	1.6s	12.58	12.3s	20.3s
coAuthorsDBLP	3.0E5	9.8E5	0.92	245.3s	250.8s	11X	26X	30.7s	**0.49**	15.7s	**26.5s**	10X	4X	3.0s	1.26	255.3s	275.0s
coAuthorsCite	2.2E5	8.1E5	0.49	77.0s	81.3s	11X	33X	8.2s	**0.41**	5.4s	**13.3s**	10X	7X	2.1s	1.01	81.1s	90.4s
citationCite	2.6E5	1.1E6	**0.48**	2.0E3s	2.1E3s	13X	27X	32.3s	0.52	3.5s	**24.8s**	11X	2X	5.2s	0.86	288.1s	314.0s
coPapersDBLP	5.4E5	1.5E7	*NA*	*NA*	*NA*	13X	210X	52.8s	**0.14**	17.4s	**61.6s**	15X	13X	27.8s	0.78	775.2s	919.5s
coPapersCite	4.3E5	1.6E7	*NA*	*NA*	*NA*	32X	950X	16.4s	**0.11**	0.9s	**51.6s**	29X	39X	26.0	0.37	72.7s	210.6s
appu*	1.4E4	9.2E5	**21.70**	250.0s	250.1s	5X	117X	25.5s	22.40	0.5s	**7.5s**	4X	1.2X	3.0s	27.69	15.9s	24.5s

Table 3: Mean relative errors for the first 10 and 40 eigenvalues.

Graph	r	k=10					k=40				
		loc. (edge)	loc. (neig.)	heav. edge	Kron	ours	loc. (edge)	loc. (neig.)	heav. edge	Kron	ours
airfoil	70%	1.05	0.93	4.74	1.99	**0.46**	0.88	0.84	2.27	2.08	**0.48**
yeast	70%	3.50	0.41	3.39	1.87	**0.31**	2.18	0.45	2.50	1.95	**0.32**
bunny	70%	**0.08**	0.32	0.13	1.81	0.16	**0.10**	0.30	0.13	1.19	0.33
minnesota	70%	4.58	1.87	9.30	1.95	**0.34**	2.11	1.61	4.16	2.09	**0.32**

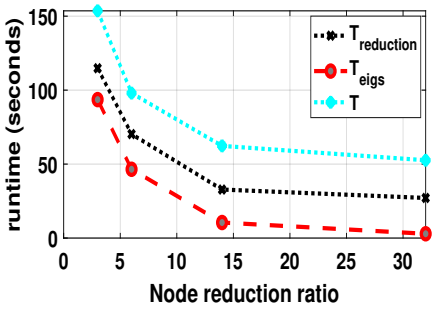

Figure 9: Runtime for spectral clustering under different reduction ratios for the "coPapersCiteseer" [8].

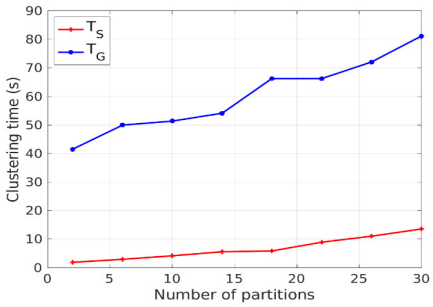

Figure 10: Runtime for graph clustering with different number of clusters for the "coAuthorsCiteseer" [8].

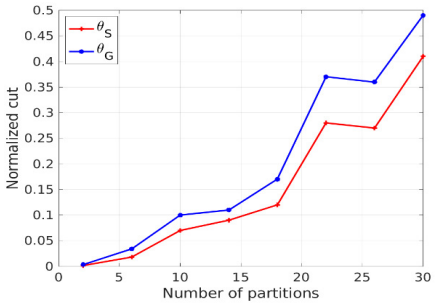

Figure 11: Normalized cut for graph clustering with different number of clusters for the "coAuthorsCiteseer" [8].

Table 4: The number of edge comparison.

Graph	loc. (edge)	loc. (neig.)	heav.edge	Kron	ours
airfoil	3,126	3,246	3,322	589,487	**1,049**
yeast	713	779	603	60,806	**390**
bunny	8,897	11,059	8,838	280,875	**981**
minnesota	1,264	1,259	**603**	3,675	732

promising performance in efficiency and reliability achieved by the proposed algorithm.

We also evaluated the performance of the spectral clustering algorithm using the original graph and the spectrally-coarsened graph under different numbers of clusters. As shown in Figure 10 and Figure 11, the coarsened graph has 11× fewer nodes and 26× fewer edges compare to the original graph. And T_G and T_S are the total clustering time when using the original graph and the coarsened graph. With the increasing number of partitions, we observed that the spectral clustering method using the spectrally-coarsened graph is much faster with consistently higher partitioning qualities.

6.4 Results of scalable data visualization

We first demonstrate the connection between the t-SNE embedding solution and the first few unnormalized Laplacian eigenvectors of the k-NN graph formed with the original data set. We increase the number of Laplacian eigenvectors for representing the embedding vectors $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^n$ that store the locations of n data points in 2D space obtained by running t-SNE, and compute the correlation factors $p^x_{tsne} = \frac{||\mathbf{U}\mathbf{U}^\top \mathbf{x}||_2}{||\mathbf{x}||_2}$ and $p^y_{tsne} = \frac{||\mathbf{U}\mathbf{U}^\top \mathbf{y}||_2}{||\mathbf{y}||_2}$, where $\mathbf{U} \in \mathbb{R}^{n \times r}$ is the matrix with the first r Laplacian eigenvectors (of the original k-NN graph) as its column vectors. If p^x_{tsne} or p^y_{tsne} is close to 1, it indicates a strong correlation (significant overlap) between the eigenspace and the t-SNE embedding vectors. Figure 12 shows strong correlations between the low-dimensional embedding

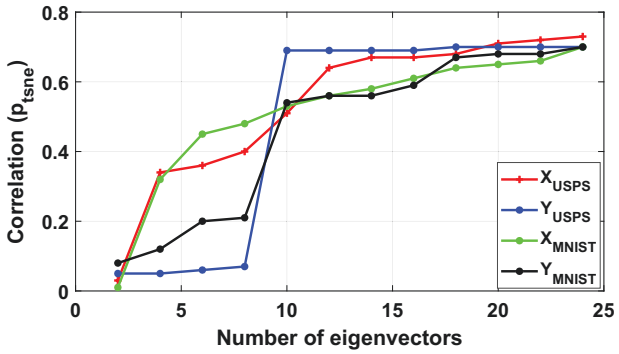

Figure 12: Correlations between 2D embedding vectors computed by t-SNE and the subspace formed by the first few eigenvectors.

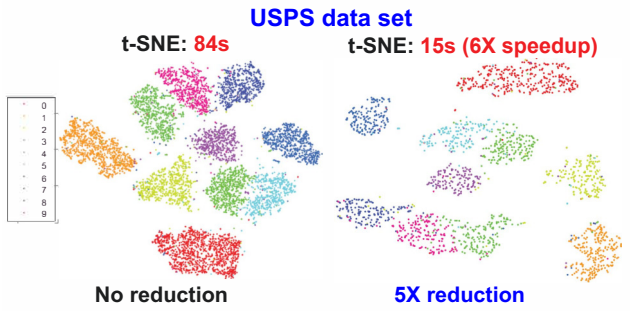

Figure 13: t-SNE visualization with original USPS data set and the reduced data set.

vectors and the first few (e.g. $r = 20$) eigenvectors of the Laplacian matrices corresponding to the k-NN graphs constructed using the USPS and MNIST data sets [3], where X_{USPS} and X_{MNIST} represent p^x_{tsne}; Y_{USPS} and Y_{MNIST} represent p^y_{tsne}. It is also interesting to observe that the t-SNE embedding vectors are more closely related to the 10-th eigenvector, since the inclusion of such an eigenvector leads to significantly improved correlation factors p^x_{tsne} and p^y_{tsne}. This is actually very reasonable considering the ground-truth number of clusters for the USPS and MNIST data sets is 10.

We demonstrate the t-SNE visualization results on the original and reduced USPS and MNIST data sets obtained by leveraging spectrally-coarsened NN graphs in Figure 13 and Figure 14. Our results show very clear cluster boundaries after spectral graph coarsening, which retain the ones obtained from the original data sets, indicating very high-quality embedding results as well as significantly improved runtime performance.

To better show the scalability of this framework, we choose to apply it to a larger YouTube social network [4] with more than one million data and 5,000 categories (communities). Every node is labeled with the communities it belongs to, if it is one of the most

[3] **USPS** includes 9, 298 images of USPS hand written digits with 256 attributes; **MNIST** is a data set from Yann LeCun's website http://yann.lecun.com/exdb/mnist/, which includes 70, 000 images of hand written digits with each of them represented by 784 attributes.

[4] Available at https://snap.stanford.edu/data/com-Youtube.html.

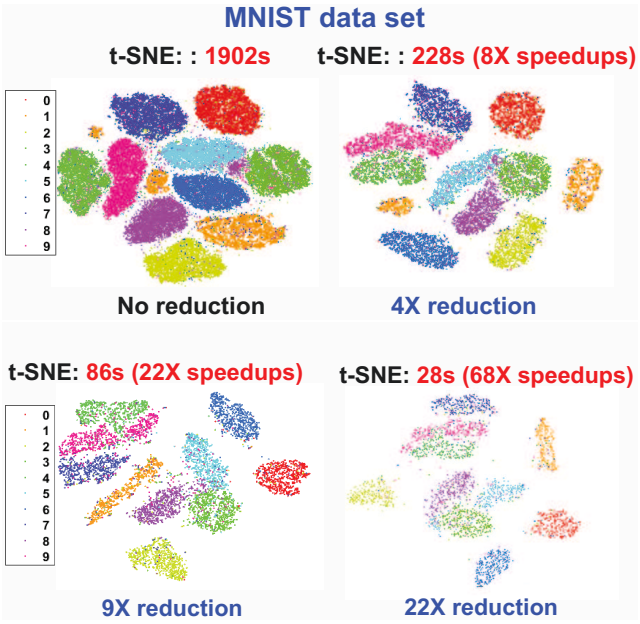

Figure 14: t-SNE visualization with original MNIST data set and data sets under different reduction ratios.

Table 5: Visualization time for two data sets

Data set	Reduction ratio	$T_{reduction}$	T_{tsne}	$T_{LargeVis}$
MNIST	(1X)	–	1,902s	838s
MNIST	(10X)	58s	86s	535s
Youtube	(1X)	–	59,222s	6,460s
Youtube	(108X)	413s	109s	546s

popular 5,000 communities, or with a special category named others. We apply both t-SNE and LargeVis [44] data visualization tools with default settings for the experiments. Table 5 shows the runtime for visualizing the MNIST and YouTube data sets, where $T_{reduction}$, T_{tsne} and $T_{LargeVis}$ represent graph reduction time, t-SNE visualization time, and LargeVis visualization time. We need to mention that all the k-NN graphs are constructed by the LargeVis tool. The data reduction ratio is also shown in the table. We can see that the framework can aggressively accelerate the data visualization for both t-SNE and LargeVis tool with satisfying accuracy preserved on the reduced data sets.

7 CONCLUSION

We propose a scalable algorithmic framework for spectral coarsening of large undirected graphs, which allows computing much smaller graphs while preserving the key spectrum of the original graph. We show that the resultant spectrally-coarsened graphs can robustly preserve the first few nontrivial eigenvalues and eigenvectors of the original graph Laplacian. In addition, the spectral graph coarsening method has been leveraged to develop much faster algorithms for multilevel spectral graph clustering as well as visualization of large data sets. We conducted extensive experiments using a variety of large graphs and data sets and obtained

very promising results. For instance, we are able to coarsen the "co-PapersCiteseer" graph with 0.43 million nodes and 16 million edges to a much smaller graph with only $13K$ (32X fewer) nodes and $17K$ (950X fewer) edges in about 16 seconds; the spectrally-coarsened graphs also allow us to achieve up to $1,100X$ speedup for spectral graph clustering and up to $60X$ speedup for t-SNE visualization of large data sets.

8 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] D. A. Bader, H. Meyerhenke, P. Sanders, C. Schulz, A. Kappes, and D. Wagner. Benchmarking for graph clustering and partitioning. In *Encyclopedia of Social Network Analysis and Mining*, pages 73–82. Springer, 2014.

[2] D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner. Graph partitioning and graph clustering. In *10th DIMACS Implementation Challenge Workshop*, 2012.

[3] J. Batson, D. Spielman, and N. Srivastava. Twice-Ramanujan Sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012.

[4] A. A. Benczúr and D. R. Karger. Approximating st minimum cuts in õ (n 2) time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 47–55. ACM, 1996.

[5] A. A. Benczúr and D. R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *SIAM Journal on Computing*, 44(2):290–319, 2015.

[6] J. Chen and I. Safro. Algebraic distance on graphs. *SIAM Journal on Scientific Computing*, 33(6):3468–3490, 2011.

[7] P. Christiano, J. Kelner, A. Madry, D. Spielman, and S. Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proc. ACM STOC*, pages 273–282, 2011.

[8] T. Davis and Y. Hu. The university of florida sparse matrix collection. *ACM Trans. on Math. Soft. (TOMS)*, 38(1):1, 2011.

[9] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[10] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(11):1944–1957, 2007.

[11] F. Dorfler and F. Bullo. Kron reduction of graphs with applications to electrical networks. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(1):150–163, 2012.

[12] M. Elkin and D. Peleg. Approximating k-spanner problems for k> 2. *Theoretical Computer Science*, 337(1-3):249–277, 2005.

[13] Z. Feng. Spectral graph sparsification in nearly-linear time leveraging efficient spectral perturbation analysis. In *Design Automation Conference (DAC), 2016 53nd ACM/EDAC/IEEE*, pages 1–6. IEEE, 2016.

[14] Z. Feng. Similarity-aware spectral sparsification by edge filtering. In *Design Automation Conference (DAC), 2018 55nd ACM/EDAC/IEEE*. IEEE, 2018.

[15] W. Fung, R. Hariharan, N. Harvey, and D. Panigrahi. A general framework for graph sparsification. In *Proc. ACM STOC*, pages 71–80, 2011.

[16] D. Harel and Y. Koren. A fast multi-scale method for drawing large graphs. In *International symposium on graph drawing*, pages 183–196. Springer, 2000.

[17] G. B. Hermsdorff and L. Gunderson. A unifying framework for spectrum-preserving graph sparsification and coarsening. In *Advances in Neural Information Processing Systems*, pages 7736–7747, 2019.

[18] H. Jeong, S. P. Mason, A.-L. Barabási, and Z. N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41, 2001.

[19] G. Karypis and V. Kumar. Metis–unstructured graph partitioning and sparse matrix ordering system, version 2.0. 1995.

[20] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[21] S. Kaski and J. Peltonen. Dimensionality reduction for data visualization [applications corner]. *IEEE signal processing magazine*, 28(2):100–104, 2011.

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

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

[24] Y. T. Lee and H. Sun. An SDP-based Algorithm for Linear-sized Spectral Sparsification. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 678–687, New York, NY, USA, 2017. ACM.

[25] H. Li and A. Schild. Spectral subspace sparsification. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 385–396. IEEE, 2018.

[26] J. Liang, S. Gurukar, and S. Parthasarathy. Mile: A multi-level framework for scalable graph embedding. *arXiv preprint arXiv:1802.09612*, 2018.

[27] G. C. Linderman and S. Steinerberger. Clustering with t-sne, provably. *SIAM Journal on Mathematics of Data Science*, 1(2):313–332, 2019.

[28] O. Livne and A. Brandt. Lean algebraic multigrid (LAMG): Fast graph Laplacian linear solver. *SIAM Journal on Scientific Computing*, 34(4):B499–B522, 2012.

[29] A. Loukas. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning Research*, 20(116):1–42, 2019.

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

[31] D. Peleg and A. A. Schäffer. Graph spanners. *Journal of graph theory*, 13(1):99–116, 1989.

[32] R. Peng, H. Sun, and L. Zanetti. Partitioning well-clustered graphs: Spectral clustering works. In *Proceedings of The 28th Conference on Learning Theory (COLT)*, pages 1423–1455, 2015.

[33] R. Preis and R. Diekmann. Party-a software library for graph partitioning. *Advances in Computational Mechanics with Parallel and Distributed Processing*, pages 63–71, 1997.

[34] M. Purohit, B. A. Prakash, C. Kang, Y. Zhang, and V. Subrahmanian. Fast influence-based coarsening for large networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1296–1305, 2014.

[35] Y. Saad. *Iterative methods for sparse linear systems*, volume 82. siam, 2003.

[36] V. Satuluri, S. Parthasarathy, and Y. Ruan. Local graph sparsification for scalable clustering. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 721–732, 2011.

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

[38] D. I. Shuman, M. J. Faraji, and P. Vandergheynst. A multiscale pyramid transform for graph signals. *IEEE Transactions on Signal Processing*, 64(8):2119–2134, 2015.

[39] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[40] D. Spielman and S. 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.

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

[42] D. A. Spielman and S.-H. Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.

[43] D. A. Spielman and J. Woo. A note on preconditioning by low-stretch spanning trees. *arXiv preprint arXiv:0903.2816*, 2009.

[44] J. Tang, J. Liu, M. Zhang, and Q. Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th international conference on world wide web*, pages 287–297. International World Wide Web Conferences Steering Committee, 2016.

[45] S.-H. Teng. Scalable algorithms for data and network analysis. *Foundations and Trends® in Theoretical Computer Science*, 12(1–2):1–274, 2016.

[46] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318. ACM, 1994.

[47] L. Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

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

[49] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In *International Symposium on Graph Drawing*, pages 171–182. Springer, 2000.

[50] L. Wang, Y. Xiao, B. Shao, and H. Wang. How to partition a billion-node graph. In *2014 IEEE 30th International Conference on Data Engineering*, pages 568–579. IEEE, 2014.

[51] Z. Zhao and Z. Feng. A spectral graph sparsification approach to scalable vectorless power grid integrity verification. In *Proceedings of the 54th Annual Design Automation Conference 2017*, page 68. ACM, 2017.

[52] Z. Zhao and Z. Feng. Effective-resistance preserving spectral reduction of graphs. In *Proceedings of the 56th Annual Design Automation Conference 2019*, page 109. ACM, 2019.

[53] Z. Zhao, Y. Wang, and Z. Feng. SAMG: Sparsified graph theoretic algebraic multigrid for solving large symmetric diagonally dominant (SDD) matrices. In *Proceedings of ACM/IEEE International Conference on Computer-Aided Design*, pages 601–606, 2017.

[54] 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.