Towards Scalable Spectral Clustering via Spectrum-Preserving Sparsification

Yongyu Wang[1]
yongyuw@mtu.edu
Zhuo Feng[2]
zhuo.feng@stevens.edu

[1] Michigan Technological University
Houghton, Michigan, USA
[2] Stevens Institute of Technology
Hoboken, New Jersey, USA

Abstract

Eigenvalue decomposition of Laplacian matrices for large nearest-neighbor (NN) graphs is the major computational bottleneck in spectral clustering (SC). To fundamentally address this computational challenge in SC, we propose a scalable spectral sparsification framework that enables to construct nearly-linear-sized ultra-sparse NN graphs with guaranteed preservation of key eigenvalues and eigenvectors of the original Laplacian. The proposed method is based on the latest theoretical results in spectral graph theory and thus can be applied to robustly handle general undirected graphs. By leveraging a nearly-linear time spectral graph topology sparsification phase and a subgraph scaling phase via stochastic gradient descent (SGD) iterations, our approach allows computing tree-like NN graphs that can serve as high-quality proxies of the original NN graphs, leading to highly-scalable and accurate SC of large data sets. Our extensive experimental results on a variety of public domain data sets show dramatically improved performance when compared with state-of-the-art SC methods.

1 Introduction

Clustering is playing increasingly important roles in image processing and computer vision [2, 17]. It also has potential use in more fields such as computer system design and biomedical applications [22, 23, 25, 26, 27]. For example, [25, 26] successfully applied clustering methods for cell type identification and detection.

Among the existing clustering techniques, spectral methods have gained great attention in recent years [11]. Although spectral methods have many advantages, such as easy implementation, good clustering quality and rigorous theoretical foundations [16], the high computational cost due to the involved eigenvalue decomposition procedure can immediately hinder their applications in emerging large scale tasks [4].

Recent research efforts attempted to address the computational bottleneck of spectral clustering (SC) through various kinds of approximation methods: [9] proposed a Nyström-based approximation method; [5] proposed a landmark-based representation method; [19] proposed a k-means-based approximate spectral method (KASP) to accelerate SC by grouping the centroids; [3] proposed to accelerate SC via Maximum Spanning Tree; [15] introduced a compressive method (CSC) by using graph filtering to accelerate SC. However,

© 2022. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

none of these methods can reliably preserve the spectra of the original graphs, and thus lead to degraded clustering result. For instance, for the large scale Covtype data set, the clustering accuracy drops more than 15% with the Nyström and CSC methods, more than 20% with the Nyström and KASP method and more than 25% with the landmark-based methods. From the hardware perspective, new devices such as memristors [20, 21] are also applied to achieve efficient unsupervised learning [18].

In this paper, we propose a highly-scalable SC framework based on spectrum-preserving graph sparsification that can significantly accelerate SC without loss of accuracy. The key contributions of this work have been summarized as follows:

1. Compared with existing approximation approaches that have no guarantee of the solution quality in SC, our method can robustly preserve the most critical spectral properties of the original graph, such as the first few eigenvalues and eigenvectors of graph Laplacians, within **nearly-linear-sized tree-like subgraphs** that allows for computing high-quality clustering results.

2. A practically-efficient spectral graph topology sparsification method [7] has been applied for identifying spectrally critical edges to be kept in subgraphs, while a novel **scalable subgraph scaling scheme** via stochastic gradient descent (SGD) iterations has been proposed in this paper to scale up edge weights to further improve the spectral approximation of the subgraph.

3. We show that spectral graph sparsification can be considered as a **"low-pass" graph filter** for removing edges less critical for preserving the first few graph Laplacian eigenvectors. We also introduce a simple yet effective procedure for filtering out errors in Laplacian eigenvectors, which enables to leverage much sparser subgraphs in SC for achieving superior solution quality.

4. We conducted extensive experiments with well-known public domain data sets to show that the proposed method can dramatically improve SC efficiency without loss of accuracy.

2 Preliminaries

Spectral clustering Algorithm: Consider a similarity graph $G = (V, E_G, \mathbf{w_G})$, where V and E_G denote the graph vertex and edge sets, respectively, while $\mathbf{w}_G$ denotes a weight function that assigns positive weights to all edges. The Laplacian matrix of graph G is defined as follows:

$$\mathbf{L}_G(i,j) = \begin{cases} -w_{ij} & \text{if } (i,j) \in E_G \\ \sum_{(i,k)\in E_G} w_{ik} & \text{if } (i=j) \\ 0 & \text{otherwise .} \end{cases} \tag{1}$$

Given a set of data samples, SC find clusters with the following three steps: 1) construct a Laplacian matrix according to the similarities between data points; 2) embed nodes into k-dimensional space using the first k nontrivial graph Laplacian eigenvectors; 3) apply k-means to partition the embedded data points into k clusters.

Spectral Graph Sparsification: Graph sparsification aims to find a subgraph $S = (V, E_S, \mathbf{w}_S)$ that has the same set of vertices of the original graph $G = (V, E_G, \mathbf{w}_G)$, but much fewer edges. We say G and its subgraph S are σ-spectrally similar if the following condition holds for all real vectors $\mathbf{x} \in R^V$

$$\frac{\mathbf{x}^\top \mathbf{L}_S \mathbf{x}}{\sigma} \leq \mathbf{x}^\top \mathbf{L}_G \mathbf{x} \leq \sigma \mathbf{x}^\top \mathbf{L}_S \mathbf{x}, \tag{2}$$

where $\mathbf{L}_G$ and $\mathbf{L}_S$ denote the Laplacian matrices of G and S, respectively. By defining the relative condition number to be $\kappa(\mathbf{L}_G, \mathbf{L}_S) = \lambda_{\max}/\lambda_{\min}$, where $\lambda_{\max}$ ($\lambda_{\min}$) denotes the largest (smallest nonzero) eigenvalues of $\mathbf{L}_\mathbf{S}^+ \mathbf{L}_\mathbf{G}$, and $\mathbf{L}_\mathbf{S}^+$ denotes the Moore-Penrose pseudoinverse of $\mathbf{L}_\mathbf{S}$, it can be further shown that $\kappa(\mathbf{L}_\mathbf{G}, \mathbf{L}_\mathbf{S}) \leq \sigma^2$, indicating that a smaller relative condition number or σ^2 corresponds to a higher spectral similarity.

3 SGD-based Two-Phase Spectral Graph Sparsification

ARPACK is the standard solver for solving practical large-scale eigenvalue problems [10]. The MATLAB also uses ARPACK as its eigensolver. [2] shows that the ARPACK employs implicitly iterative restarted Arnoldi process that contains at most $(z-k)$ steps, where z is the Arnoldi length empirically set to $2k$ and k is the number of desired eigenvalues. The overall runtime cost of ARPACK solver is proportional to $O(z^3) + (O(nz) + O(nw)) \times O(z-k)$, where n is the number of data points, w is the number of nearest neighbors [4]. Algorithms with a sparsified Laplacian can reduce the time cost of the ARPACK solver due to the reduced w. Our goal is to achieve good clustering quality with a very sparse graph.

In the following, we assume that $G = (V, E_G, \mathbf{w_G})$ is a weighted undirected graph, whereas $S = (V, E_S, \mathbf{w}_S)$ is its graph sparsifier. The descending eigenvalues of $\mathbf{L}_S^+ \mathbf{L}_G$ are denoted by $\lambda_{\max} = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$, where $\mathbf{L}_S^+$ denotes the Moore-Penrose pseudoinverse of $\mathbf{L}_S$.

3.1 Phase 1: Spectral Graph Topology Sparsification

3.1.1 Off-Tree Edge Embedding with Approximate Generalized Eigenvectors

[13] showed that $\mathbf{L}_S^+ \mathbf{L}_G$ has at most k eigenvalues greater than $\mathrm{st}_S(G)/k$, where $\mathrm{st}_S(G)$ is the total stretch of the spanning-tree subgraph S with respect to the original graph G. The total stretch can be considered as the spectral distortion due to the subgraph approximation. Recent results show that every graph has a low-stretch spanning tree (LSST) with bounded total stretch [1]:

$$O(m \log m \log\log n) \geq \mathrm{st}_S(G) = \mathrm{tr}(\mathbf{L}_S^+ \mathbf{L}_G) \geq \kappa(\mathbf{L}_G, \mathbf{L}_S), \tag{3}$$

where $m = |E_G|$, $n = |V|$, and $\mathrm{tr}(\mathbf{L}_S^+ \mathbf{L}_G)$ is the trace of $\mathbf{L}_S^+ \mathbf{L}_G$. To identify the key off-tree edges to be added to the low-stretch spanning tree (LSST) for dramatically reducing spectral distortion, a spectral embedding scheme using approximate generalized eigenvectors has been introduced in [7], which is based on the following spectral perturbation analysis:

$$\mathbf{L}_G(\mathbf{u}_i + \delta\mathbf{u}_i) = (\lambda_i + \delta\lambda_i)(\mathbf{L}_S + \delta\mathbf{L}_S)(\mathbf{u}_i + \delta\mathbf{u}_i), \tag{4}$$

where a perturbation $\delta\mathbf{L}_S$ is applied to $\mathbf{L}_S$, which results in perturbations in generalized eigenvalues $\lambda_i + \delta\lambda_i$ and eigenvectors $\mathbf{u}_i + \delta\mathbf{u}_i$ for $i = 1, \ldots, n$, respectively. The first-order perturbation analysis leads to

$$-\frac{\delta\lambda_i}{\lambda_i} = \mathbf{u}_i^\top \delta\mathbf{L}_S \mathbf{u}_i, \tag{5}$$

which indicates that the reduction of λ_i is proportional to the Laplacian quadratic form of $\delta\mathbf{L}_S$ with the generalized eigenvector $\mathbf{u}_i$. (5) can also be understood through the following **Courant-Fischer theorem** for generalized eigenvalue problems:

$$\lambda_{max} = \lambda_1 = \max_{\substack{|\mathbf{x}|\neq 0 \\ \mathbf{x}^\top \mathbf{1}=0}} \frac{\mathbf{x}^\top \mathbf{L}_G \mathbf{x}}{\mathbf{x}^\top \mathbf{L}_S \mathbf{x}} \geq \max_{\substack{|\mathbf{x}|\neq 0 \\ x(p)\in\{0,1\}}} \frac{\mathbf{x}^\top \mathbf{L}_G \mathbf{x}}{\mathbf{x}^\top \mathbf{L}_S \mathbf{x}} = \max \frac{|\partial_G(Q)|}{|\partial_S(Q)|}, \tag{6}$$

where $\mathbf{1}$ is the all-one vector, the node set Q is defined as $Q \overset{\text{def}}{=} \{p \in V : x(p) = 1\}$, and the boundary of Q in G is defined as $\partial_G(Q) \overset{\text{def}}{=} \{(p,q) \in E_G : p \in Q, q \notin Q\}$, which obviously leads to $\mathbf{x}^\top \mathbf{L}_G \mathbf{x} = |\partial_G(Q)|, \mathbf{x}^\top \mathbf{L}_S \mathbf{x} = |\partial_S(Q)|$. As a result, $\lambda_{max} = \lambda_1$ is the upper bound of the largest mismatch in boundary (cut) size between G and S. Once Q or $\partial_S(Q)$ is found via graph embedding using dominant generalized eigenvectors, we can selectively pick the edges from $\partial_G(Q)$ and recover them to S to mitigate maximum mismatch or λ_1.

Denote $\mathbf{e}_p \in R^V$ the vector with only the p-th element being 1 and others being 0. We also denote $\mathbf{e}_{pq} = \mathbf{e}_p - \mathbf{e}_q$, then the generalized eigenvalue perturbation due to the inclusion of off-tree edges can be expressed as follows

$$-\frac{\delta\lambda_i}{\lambda_i} = \mathbf{u}_i^\top \delta\mathbf{L}_{S,max} \mathbf{u}_i = \sum_{(p,q)\in E_G \setminus E_S} w_{pq} \left(\mathbf{e}_{pq}^T \mathbf{u}_i\right)^2, \tag{7}$$

where $\delta\mathbf{L}_{S,max} = \mathbf{L}_G - \mathbf{L}_S$. The **spectral criticality** c_{pq} of each off-tree edge (p,q) is defined as:

$$c_{pq} = w_{pq} \left(\mathbf{e}_{pq}^T \mathbf{u}_1\right)^2 \approx w_{pq} \left(\mathbf{e}_{pq}^T \mathbf{h}_t\right)^2, \quad \mathbf{h}_t = \left(\mathbf{L}_S^+ \mathbf{L}_G\right)^t \mathbf{h}_0, \tag{8}$$

where $\mathbf{h}_t$ denotes the approximate dominant generalized eigenvector computed through a small number (e.g. $t = 2$) of generalized power iterations using an initial random vector $\mathbf{h}_0$.

3.1.2 A Trace Minimization Perspective for Spectral Sparsification.

Denote the descending eigenvalues and the corresponding unit-length, mutually-orthogonal eigenvectors of $\mathbf{L}_G$ by $\zeta_1 \geq \cdots > \zeta_n = 0$, and $\omega_1, \cdots, \omega_n$, respectively. Similarly denote the eigenvalues and eigenvectors of $\mathbf{L}_S$ by $\tilde{\zeta}_1 \geq \cdots > \tilde{\zeta}_n = 0$ and $\tilde{\omega}_1, \cdots, \tilde{\omega}_n$, respectively. It should be noted that both ω_n and $\tilde{\omega}_n$ are the normalized and orthogonal to the all-one vector $\mathbf{1}/\sqrt{n}$. Then the following spectral decompositions of $\mathbf{L}_G$ and $\mathbf{L}_S^+$ always hold:

$$\mathbf{L}_G = \sum_{i=1}^{n-1} \zeta_i \omega_i \omega_i^\top, \qquad \mathbf{L}_S^+ = \sum_{j=1}^{n-1} \frac{1}{\tilde{\zeta}_j} \tilde{\omega}_j \tilde{\omega}_j^\top, \tag{9}$$

which leads to the following trace of $\mathbf{L}_S^+ \mathbf{L}_G$:

$$\mathbf{Tr}(\mathbf{L}_S^+ \mathbf{L}_G) = \mathbf{Tr}(\sum_{i=1}^{n-1} \sum_{j=1}^{n-1} \frac{\zeta_i}{\tilde{\zeta}_j} \varepsilon_{ij} \tilde{\omega}_j \omega_i^\top) = \sum_{j=1}^{n-1} \frac{1}{\tilde{\zeta}_j} \sum_{i=1}^{n-1} \zeta_i \varepsilon_{ij}^2, \tag{10}$$

where ε_{ij} satisfies: $0 \leq \varepsilon_{ij}^2 = (\omega_i^\top \tilde{\omega}_j)^2 \leq 1$. According to (10), the most spectrally critical off-tree edges identified by (8) will impact the largest eigenvalues of $\mathbf{L}_S^+ \mathbf{L}_G$ as well as the bottom (smallest nonzero) eigenvalues of $\mathbf{L}_S$, since the smallest $\tilde{\zeta}_j$ values directly contribute to the largest components in the trace of $\mathbf{L}_S^+ \mathbf{L}_G$. This fact enables to recover small portions of most spectrally critical off-tree edges to LSST subgraph for preserving the key spectral graph properties within the sparsified graph.

3.1.3 A Scheme for Eigenvalue Stability Checking.

We propose a novel method for checking the stability of bottom eigenvalues of the sparsified Laplacian. Our approach proceeds as follows: 1) in each iteration for recovering off-tree edges, we compute and record the several smallest eigenvalues of the latest sparsified Laplacian: for example, the bottom k eigenvalues that are critical for spectral clustering tasks; 2) we determine whether more off-tree edges should be recovered by looking at the stability by comparing with the eigenvalues computed in the previous iteration: if the change of eigenvalues is significant, more off-tree edges should be added to the current sparsifier. More specifically, we store the bottom k eigenvalues computed in the previous (current) iteration into vector v_p (v_{p+1}), and calculate the eigenvalue variation ratio by:

$$ratio_{var} = \frac{\|(v_p - v_{p+1})\|}{\|(v_p)\|}. \tag{11}$$

A greater eigenvalue variation ratio indicates less stable eigenvalues within the latest sparsified graph Laplacian, and thus justifies another iteration to allow adding more "spectrally-critical" off-tree edges into the sparsifier.

3.2 Phase 2: Subgraph Scaling via SGD Iterations

To aggressively limit the number of edges in the subgraph S while still achieving a high quality approximation of the original graph G, we introduce an efficient edge scaling scheme to mitigate the accuracy loss. We propose to scale up edge weights in the subgraph S to further reduce the largest mismatch or λ_1. The dominant eigenvalue perturbation $\delta\lambda_1$ in terms of edge weight perturbations using first-order analysis (4) can be expressed as:

$$-\frac{\delta\lambda_1}{\lambda_1} = \mathbf{u}_1^\top \delta\mathbf{L}_S\mathbf{u}_1 = \sum_{(p,q)\in E_S} \delta w_{pq} \left(\mathbf{e}_{pq}^\top \mathbf{u}_1\right)^2, \tag{12}$$

which directly gives the sensitivity of λ_1 with respect to each edge weight w_{pq} as follows:

$$\frac{\delta\lambda_1}{\delta w_{pq}} = -\lambda_1 \left(\mathbf{e}_{pq}^\top \mathbf{u}_1\right)^2 \approx -\lambda_1 \left(\mathbf{e}_{pq}^\top \mathbf{h}_t\right)^2. \tag{13}$$

With the (approximate) sensitivity expressed in (13), it is possible to find a proper weight scaling factor for each edge in S such that λ_1 will be dramatically reduced. However, since both λ_1 and λ_n will decrease monotonically when scaling up edge weights, it is likely that λ_n will decrease at a faster rate than λ_1, which will lead to even a worse spectral approximation. To address this issue, we adopt nearly-linear time algorithms for estimating the extreme generalized eigenvalues λ_1 and λ_n introduced in [8], which allows us to scale edge weight properly without degrading the spectral approximation quality. Then the following constrained nonlinear optimization framework can be leveraged for scaling up the subgraph (S) edge weights $\mathbf{w}_s$ to minimize the largest mismatch reflected by the largest generalized eigenvalue λ_1 in (6).

$$\begin{aligned}
&\textbf{minimize:} \quad \lambda_1(\mathbf{w}_s) \\
&\textbf{subject to:} \\
&\quad \textbf{(a)} \quad \mathbf{L}_G\mathbf{u}_i = \lambda_i\mathbf{L}_S\mathbf{u}_i, \;\; i = 1,...,n; \\
&\quad \textbf{(b)} \quad \lambda_{max} = \lambda_1 \geq \lambda_2 ... \geq \lambda_n = \lambda_{min}; \\
&\quad \textbf{(c)} \quad \lambda_n \;\; \geq \lambda_n^{(0)}\overline{\Delta}_{\lambda_n}.
\end{aligned} \tag{14}$$

Algorithm 1 Subgraph Edge Weight Scaling via Constrained SGD Iterations

Input: $\mathbf{L}_G$, $\mathbf{L}_S$, $\mathbf{d}_G$, $\mathbf{d}_S$, $\lambda_1^{(0)}$, $\lambda_n^{(0)}$, $\overline{\Delta}_{\lambda_n}$, β, η_{max}, ε, and N_{max}
Output: $\tilde{\mathbf{L}}_S$ with scaled edge weights

Initialize: $k=1$, $\eta^{(1)}=\eta_{max}$, $\Delta_{\lambda_n}=\left(\overline{\Delta}_{\lambda_n}\right)^{\frac{1}{N_{max}}}$, $\lambda_1^{(1)}=\lambda_1^{(0)}$, $\lambda_n^{(1)}=\lambda_n^{(0)}$, $\Delta\lambda_1^{(1)}=\lambda_1$;
Do initial subgraph edge scaling by $w_{pq}^{(1)}=\frac{w_{pq}^{(0)}\sqrt{\lambda_1^{(0)}/\lambda_n^{(0)}}}{10}$ for each edge $(p,q)\in E_S$;
while $\left(\frac{\Delta\lambda_1^{(k)}}{\lambda_1^{(k)}}\geq\varepsilon\right)\wedge(k\leq N_{max})$ **do**
  Compute approximate eigenvector $\mathbf{h}_t^{(k)}$ by (8);
  for each edge $(p,q)\in E_S$ **do**
    $s_{pq}^{(k)}:=-\lambda_1^{(k)}\left(\mathbf{e}_{pq}^{\top}\mathbf{h}_t^{(k)}\right)^2$, $\Delta w_{pq}^{(k+1)}:=\beta\Delta w_{pq}^{(k)}-\eta^{(k)}s_{pq}^{(k)}$;
    $\phi(p):=\frac{\mathbf{d}_G(p)}{\mathbf{d}_S(p)+\Delta w_{pq}^{(k+1)}}$, $\phi(q):=\frac{\mathbf{d}_G(q)}{\mathbf{d}_S(q)+\Delta w_{pq}^{(k+1)}}$;
    if $\min\left(\phi(p),\phi(q)\right)\leq\lambda_n^{(k)}\Delta_{\lambda_n}$
      $\Delta w_p:=\frac{\mathbf{d}_G(p)}{\Delta_{\lambda_n}}-\mathbf{d}_S(p)$, $\Delta w_q:=\frac{\mathbf{d}_G(q)}{\Delta_{\lambda_n}}-\mathbf{d}_S(q)$, $\Delta w_{pq}^{(k+1)}:=\min\left(\Delta w_p,\Delta w_q\right)$;
    end if
    $w_{pq}:=w_{pq}+\Delta w_{pq}^{(k+1)}$, $\mathbf{d}_S(p):=\mathbf{d}_S(p)+\Delta w_{pq}^{(k+1)}$, $\mathbf{d}_S(q):=\mathbf{d}_S(q)+\Delta w_{pq}^{(k+1)}$;
  end for
  $\eta^{(k+1)}:=\frac{\lambda_1^{(k)}}{\lambda_1}\eta_{max}$, $k:=k+1$, and update $\lambda_1^{(k)}$ & $\lambda_n^{(k)}$;
  $\Delta\lambda_1^{(k)}:=\lambda_1^{(k)}-\lambda_1^{(k-1)}$;
end while
Return $\tilde{\mathbf{L}}_S$.

In (14), $\lambda_n^{(0)}$ and λ_n denote the smallest nonzero eigenvalues before and after edge scaling, respectively, whereas $\overline{\Delta}_{\lambda_n}$ denotes the upper bound of reduction factor in $\lambda_n^{(0)}$ after edge scaling. (14) aims to minimize λ_1 by scaling up subgraph edge weights while limiting the decrease in λ_n.

To efficiently solve (14), in this paper we propose a constrained SGD algorithm with momentum [14] for iteratively scaling up edge weights, as shown in Algorithm 1. The algorithm inputs include: the graph Laplacians $\mathbf{L}_G$ and $\mathbf{L}_S$, vectors $\mathbf{d}_G$ and $\mathbf{d}_S$ for storing diagonal elements in Laplacians, the largest and smallest generalized eigenvalues $\lambda_1^{(0)}$ and $\lambda_n^{(0)}$ before edge scaling, the upper bound reduction factor $\overline{\Delta}_{\lambda_n}$ for λ_n, the coefficient β for combining the previous and the latest updates during each SGD iteration with momentum, the maximum step size η_{max} for update, as well as the SGD convergence control parameters ε and N_{max}.

The key steps of the algorithm include: **1)** a random vector is first generated and used to compute the approximate dominant eigenvector with (8) as well as edge weight sensitivity with (13); **2)** the weight update for each edge in the subgraph is estimated based on the previous update (momentum) as well as the latest step size and gradient; **3)** the impact on the reduction of λ_n will be evaluated for each weight update to make sure the decreasing rate is not too fast; **4)** check the cost function λ_1 or the largest weight sensitivity to determine whether another SGD iteration is needed.

Since edge weights in the subgraph ($\mathbf{w}_S$) will be updated during each SGD iteration, we need to solve a new subgraph Laplacian matrix $\mathbf{L}_S$ for updating the approximate eigenvector $\mathbf{h}_t$ in (13), which can be efficiently achieved by leveraging recent sparsified algebraic multi-

grid (SAMG) algorithm that has shown highly scalable performance for solving large graph Laplacians [24]. Since the subgraph topology remains unchanged during the SGD iterations, it will also be possible to exploit incremental update of graph Laplacian solvers to further improve efficiency.

4 Filtering Eigenvectors of Sparsified Laplacians

There is a clear analogy between traditional signal processing or classical Fourier analysis and graph signal processing [12]: 1) the signals at different time points in classical Fourier analysis correspond to the signals at different nodes in an undirected graph; 2) the more slowly oscillating functions in time domain correspond to the graph Laplacian eigenvectors associated with lower eigenvalues or the more slowly varying (smoother) components across the graph.

Inspired by [12], in this paper we introduce a simple yet effective procedure for filtering out errors in Laplacian eigenvectors computed using spectrally sparsified graphs, which enables to leverage ultra-sparse subgraphs in SC for achieving superior solution quality. In the following analysis, we assume that the k smallest eigenvalues and their eigenvectors of $\mathbf{L}_G$ have been pretty well preserved in $\mathbf{L}_S$ through the proposed two-phase spectral sparsification approach, while the remaining $n-k$ higher eigenvalues and eigenvectors are not. Then the spectral decompositions of $\mathbf{L}_G$ and $\mathbf{L}_S$ of (9) can be written as:

$$\begin{aligned}\mathbf{L}_G &= \sum_{i=1}^{n-1} \zeta_i \omega_i \omega_i^\top, \\ \mathbf{L}_S &= \sum_{i=1}^{n} \tilde{\zeta}_i \tilde{\omega}_\mathbf{i} \tilde{\omega}_\mathbf{i}^\top \approx \sum_{i=n-k+1}^{n} \zeta_i \omega_\mathbf{i} \omega_i^\top + \sum_{i=1}^{n-k} \tilde{\zeta}_i \tilde{\omega}_\mathbf{i} \tilde{\omega}_\mathbf{i}^\top.\end{aligned} \tag{15}$$

In the following, we show that using existing sparse eigenvalue decomposition methods for computing the first few Laplacian eigenvectors using spectrally sparsified graphs will introduce errors expressed as linear combination of eigenvectors corresponding to only large eigenvalues. Since the power iteration method is well known for calculating a few extreme eigenvalues and eigenvectors, we analyze the error introduced by sparsified Laplacians in power iterations, while the errors by other algorithms can be similarly analyzed in probably more complicated ways. To compute the smallest eigenvalues and their eigenvectors for $\mathbf{L}_G$, a few inverse power iterations can be applied:

$$(\mathbf{L}_G + z\mathbf{I})^l \mathbf{x} = \mathbf{r}^\perp, \tag{16}$$

where l is the number of inverse power iterations, $\mathbf{r}^\perp \in R^n$ is a random vector orthogonal to the all-one vector $\mathbf{1}$, z is a small positive real number, and $\mathbf{I} \in R^{n\times n}$ is an identity matrix. $z\mathbf{I}$ is added to the Laplacian matrix to make sure that the resultant matrix is non-singular.

Let $\mathbf{x}$ and $\tilde{\mathbf{x}}$ denote the true and approximate solutions obtained with $\mathbf{L}_G$ and $\mathbf{L}_S$, respectively, then we have:

$$\mathbf{x} = \sum_{i=1}^{n} \frac{\omega_i \omega_i^\top \mathbf{r}^\perp}{(\zeta_i + z)^l}; \qquad \tilde{\mathbf{x}} \approx \sum_{i=1}^{n-k} \frac{\tilde{\omega}_i \tilde{\omega}_i^\top \mathbf{r}^\perp}{(\tilde{\zeta}_i + z)^l} + \sum_{i=n-k+1}^{n} \frac{\omega_i \omega_i^\top \mathbf{r}^\perp}{(\zeta_i + z)^l}, \tag{17}$$

which allows us to express the error vector $\mathbf{e}$ as:

$$\mathbf{e} = \mathbf{x} - \tilde{\mathbf{x}} \approx \sum_{i=1}^{n-k} \left(\frac{\omega_i \omega_i^\top \mathbf{r}^\perp}{(\zeta_i + z)^l} - \frac{\tilde{\omega}_i \tilde{\omega}_i^\top \mathbf{r}^\perp}{(\tilde{\zeta}_i + z)^l} \right), \tag{18}$$

which shows that when using power iterations to compute the smallest nonzero eigenvalue and its eigenvector using sparsified graph Laplacians, the error in the eigenvector can be expressed as a linear combination of the eigenvectors corresponding to relatively large eigenvalues. If we consider each eigenvector as a signal on the graph, then the eigenvectors of large eigenvalues can be considered as highly oscillating or high frequency signals on graphs. Therefore, the error due to the sparsified graph Laplacian in inverse power iterations can be considered as a combination of high frequency signals on graphs, which thus can be efficiently filtered out using "low-pass" graph signal filters. In fact, weighted Jacobi or Gauss-Seidel methods can be efficiently applied for filtering out such high frequency error signals on graphs, which have been widely adopted in modern iterative methods for solving large sparse matrices, such as the smoothing (relaxation) function in multigrid algorithms. This work adopts a weighted Jacobi iteration scheme for filtering eigenvectors on the graph, while the detailed filtering algorithm has been described in Algorithm 2.

Algorithm 2 Algorithm for Iterative Eigenvector Filtering

Input: $\mathbf{L}_G = \mathbf{D}_G - \mathbf{A}_G$, $\tilde{\omega}_n$,..., $\tilde{\omega}_{n-k+1}$, γ,N_{filter}
Output: The smoothed eigenvectors.

1: For each of the approximate eigenvectors $\tilde{\omega}_1$,..., $\tilde{\omega}_k$, do
2: **for** $i = 1$ **to** N_{filter} **do**
3: $\quad \tilde{\omega}_j^{(i+1)} = (1-\gamma)\tilde{\omega}_j^{(i)} + \gamma(D_G - \tilde{\zeta}_j I)^{-1}\mathbf{A}_G\tilde{\omega}_j^{(i)}$
4: **end for**
5: Return the smoothed eigenvectors $\tilde{\omega}_n$,..., $\tilde{\omega}_{n-k+1}$.

5 Algorithm Flow and Complexity Analysis

The key steps of the proposed method and their computational complexities are summarized as follows: 1) Identify the edges to be kept in the subgraph using the spectral graph topology sparsification methods [2] in $O(m\log n)$ time; 2) Scale up edge weights in the subgraph using the proposed iterative SGD scheme (Algorithm 1) in $O(m)$ time; 3) Perform eigenvalue decomposition for the sparsified graph Laplacian to find the first few eigenvectors. For ultra-sparse Laplacian matrices with tree-like graphs, popular Krylov subspace iterative methods can be highly-scalable in practice due to the nearly-linear time cost for sparse matrix vector multiplications that dominate the overall computation time; 4) Apply the proposed eigenvector filtering scheme (Algorithm 2) to improve the approximation of Laplacian eigenvectors in $O(m)$ time; 5) Run the k-means algorithm on eigenvectors in $O(knd)$ time, where k is the number of clusters, and d is the dimension of the feature vectors of data points.

6 Experimental Evaluation

Experiment Setup: Experiments are performed using MATLAB running on a PC with a 2.50 GHz Intel Core i5 CPU and 8 GB RAM. The following real-world data sets are used in our experiments: **COIL-20** includes 1,440 gray scale images of 20 different objects and each image is represented by $1,024$ attributes; **PenDigits** includes $7,474$ handwritten digits and each digit is represented by 16 attributes; **USPS** includes $9,298$ images of USPS hand

written digits with 256 attributes; **MNIST** includes 70,000 images of hand written digits with each of them represented by 784 attributes; **Covtype** includes 581,012 instances for predicting forest cover type from cartographic variables and each instance with 54 attributes is from one of seven classes. We compare our method against the following state-of-the-art algorithms: (1) the **Original SC** [4], (2) the **Nyström method** [9], (3) the landmark-based methods that uses k-means for landmark selection (**LSCK**) and uses random sampling for landmark selection (**LSCR**) [5], (4) the **KASP method** [19], and (5) the **CSC method** [15]. The code can be downloaded from their authors' websites and we follow their parameter settings.

Parameter Selection: The off-tree edge budget b measures the amount of off-tree edges added to the LSST for the spectral graph topology sparsification phase, which is defined as $b = \frac{|E_S|-|V|+1}{|V|}$. b is set to be less than 0.15 for all the data sets. The number of generalized power iterations for spectral embedding is set to be $t = 2$. The parameters for the edge scaling (Algorithm 1) are: $\overline{\Delta}_{\lambda_n} = 0.5$, $\beta = 0.5$, $\eta_{max} = 0.2$, $\varepsilon = 0.01$, and $N_{max} = 100$. The parameters for the eigenvector filtering (Algorithm 2) are: $\gamma = 0.7$, and $N_{filter} = 10$. Clustering results are evaluated by comparing the cluster-memberships generated by algorithms with the ground-truth cluster-memberships provided by the data set. The accuracy metric is used [9, 5].

Experimental Results: Table 1 shows the runtime, ACC results, as well as the λ_{max} that was defined in (6). For large data sets such as **MNIST** and **Covtype**, our method achieved **1,256X** and **4,500X** times speedup, respectively. For **Covtype**, the proposed method achieves a significantly better accuracy level than any of the other approximate SC methods. Such a high quality is mainly due to the guaranteed preservation of key Laplacian eigenvalues and eigenvectors (which can be reflected in λ_{max}). On the other hand, without robust preservation of the original graph spectra, the SC results of existing approximate SC methods can be quite unsatisfactory. For example, the KASP, LSCK, LSCR and CSC algorithms that use simple sampling methods, such as k-means and random sampling, may lead to substantial loss of structural information of the data set, and thus poor accuracy in SC tasks; the approximation quality of the Nyström method strongly depends on the encoding power of the data points chosen from k-means clustering [5], but for large data set it is very unlikely that the small amount chosen data points can truthfully encode the entire data set. The efficiency of CSC is even worse than standard SC for medium-scale data set due to the following reasons: 1) it uses the dichotomy and eigenvalue count technique to estimate eigenvalue, and 2) it needs to calculate Jackson-Chebyshev polynomial for all clusters in each iteration. The accuracy of CSC is also not as robust as our method as a lot of information is lost in its random sampling and interpolation steps. We also observe that the ACC results obtained by using spectrally sparsified graphs are even better than the results obtained by using the original graphs for most data sets, indicating that our method does help remove spurious edges, leading to improved graph structures for SC. The ACC results with respect to different options of off-tree edge budget (b) have been shown in Figure 1. As observed, adding only a very small amount of off-tree edges will suffice for achieving the peak ACC result and even with much greater number of off-tree edges added into the LSST, the sparsified graphs without edge scaling still can not reach the ACC level achieved by using edge scaling.

	Clustering Accuracy (ACC)							Spectral Clustering Time							
Data Set	Orig	Nyström	KASP	LSCK	LSCR	CSC	Ours	Orig	Nyström	KASP	LSCK	LSCR	CSC	Ours	λ_{max}
COIL-20	78.80	67.44	58.83	72.41	68.45	75.83	76.27	0.37	0.46	2.74	2.44	0.23	1.57	0.28 (1.32X)	138
PenDigits	81.12	68.70	75.83	80.77	77.89	47.09	83.26	0.47	0.28	1.00	0.81	0.23	6.03	0.36 (1.30X)	230
USPS	68.22	68.83	72.61	77.54	66.22	66.53	70.74	1.02	0.40	6.88	7.08	0.24	7.02	0.30 3.40X)	437
MNIST	71.95	53.27	68.03	69.88	57.24	29.86	72.27	6785	0.80	754	722	0.81	174.29	5.40 (**1,256X**)	569
Covtype	48.83	24.78	27.11	22.80	22.79	32.74	48.86	91,504	18.51	1,165	1,154	7.23	594.82	20.33 (**4,500X**)	456

Table 1: Clustering accuracy (%) and clustering time (seconds)

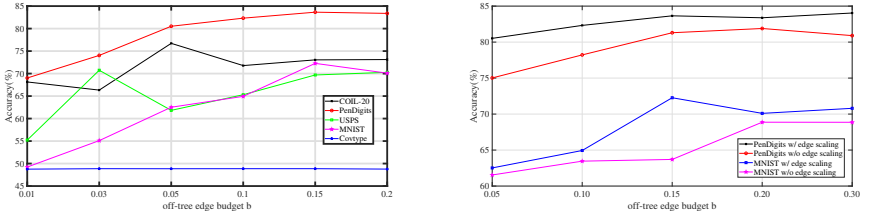

Figure 1: Left: ACC VS off-tree edge budget b; Right: ACC w/ and w/o edge scaling.

7 Conclusions

To fundamentally address the computational challenge due to the eigen-decomposition step in SC, this work introduces a novel scalable SC framework that enables to construct a very sparse sparsifier with guaranteed preservation of the original spectrum for SC purpose. Our results on a variety of public domain data sets show dramatically improved clustering performance when compared with state-of-the-art SC methods.

References

[1] Ittai Abraham and Ofer Neiman. Using petal-decompositions to build a low stretch spanning tree. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 395–406. ACM, 2012.

[2] Francis Bach and Michael Jordan. Learning spectral clustering. *Advances in neural information processing systems*, 16(2):305–312, 2004.

[3] Aditya Challa, Sravan Danda, BS Sagar, and Laurent Najman. Power spectral clustering. *Journal of Mathematical Imaging and Vision*, 62(9):1195–1213, 2020.

[4] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y Chang. Parallel spectral clustering in distributed systems. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):568–586, 2011.

[5] Xinlei Chen and Deng Cai. Large scale spectral clustering with landmark-based representation. In *AAAI*, 2011.

[6] Anna Choromanska, Tony Jebara, Hyungtae Kim, Mahesh Mohan, and Claire Monteleoni. Fast spectral clustering via the nyström method. In *International Conference on Algorithmic Learning Theory*, pages 367–381. Springer, 2013.

[7] Zhuo 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.

[8] Zhuo Feng. Similarity-aware spectral sparsification by edge filtering. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2018.

[9] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the nystrom method. *IEEE transactions on pattern analysis and machine intelligence*, 26(2):214–225, 2004.

[10] RB Lehoucq, DC Sorensen, and C Yang. Arpack users' guide: Solution of large scale eigenvalue problems with implicitly restarted arnoldi methods. *Software Environ. Tools*, 6, 1997.

[11] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[12] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre 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.

[13] Daniel A Spielman and Jaeoh Woo. A note on preconditioning by low-stretch spanning trees. *arXiv preprint arXiv:0903.2816*, 2009.

[14] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[15] Nicolas Tremblay, Gilles Puy, Rémi Gribonval, and Pierre Vandergheynst. Compressive spectral clustering. In *International Conference on Machine Learning*, pages 1002–1011, 2016.

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

[17] Xia Wan and C-CJ Kuo. A new approach to image retrieval with hierarchical color clustering. *IEEE transactions on circuits and systems for video technology*, 8(5):628–643, 1998.

[18] Zhongrui Wang, Saumil Joshi, Sergey Savel'ev, Wenhao Song, Rivu Midya, Yunning Li, Mingyi Rao, Peng Yan, Shiva Asapu, Ye Zhuo, et al. Fully memristive neural networks for pattern classification with unsupervised learning. *Nature Electronics*, 1 (2):137–145, 2018.

[19] Donghui Yan, Ling Huang, and Michael I Jordan. Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 907–916. ACM, 2009.

[20] J Joshua Yang, Matthew D Pickett, Xuema Li, Douglas AA Ohlberg, Duncan R Stewart, and R Stanley Williams. Memristive switching mechanism for metal/oxide/metal nanodevices. *Nature nanotechnology*, 3(7):429–433, 2008.

[21] J Joshua Yang, Dmitri B Strukov, and Duncan R Stewart. Memristive devices for computing. *Nature nanotechnology*, 8(1):13–24, 2013.

[22] Junyao Yang, Yuchen Wang, and Zhenlin Wang. Efficient modeling of random sampling-based lru. In *50th International Conference on Parallel Processing*, pages 1–11, 2021.

[23] Weiming Zhao and Zhenlin Wang. Scaleupc: a upc compiler for multi-core systems. In *Proceedings of the Third Conference on Partitioned Global Address Space Programing Models*, pages 1–8, 2009.

[24] 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.

[25] Ruiqing Zheng, Min Li, Zhenlan Liang, Fang-Xiang Wu, Yi Pan, and Jianxin Wang. Sinnlrr: a robust subspace clustering method for cell type detection by non-negative and low-rank representation. *Bioinformatics*, 35(19):3642–3650, 2019.

[26] Ruiqing Zheng, Zhenlan Liang, Xiang Chen, Yu Tian, Chen Cao, and Min Li. An adaptive sparse subspace clustering for cell type identification. *Frontiers in genetics*, 11:407, 2020.

[27] Ruiqing Zheng, Zhenlan Liang, Xiangmao Meng, Yu Tian, and Min Li. A robust single cell clustering method based on subspace learning and partial imputation. In *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 140–145. IEEE, 2020.