diGRASS: Directed Graph Spectral Sparsification via Spectrum-Preserving Symmetrization

YING ZHANG, ZHIQIANG ZHAO, and ZHUO FENG, Stevens Institute of Technology, USA

Recent *spectral graph sparsification* research aims to construct ultra-sparse subgraphs for preserving the original graph spectral (structural) properties, such as the first few Laplacian eigenvalues and eigenvectors, which has led to the development of a variety of nearly-linear time numerical and graph algorithms. However, there is very limited progress for spectral sparsification of directed graphs. In this work, we prove the existence of nearly-linear-sized spectral sparsifiers for directed graphs under certain conditions. Furthermore, we introduce a practically-efficient spectral algorithm (diGRASS) for sparsifying real-world, large-scale directed graphs leveraging spectral matrix perturbation analysis. The proposed method has been evaluated using a variety of directed graphs obtained from real-world applications, showing promising results for solving directed graph Laplacians, spectral partitioning of directed graphs, and approximately computing (personalized) PageRank vectors.

CCS Concepts: • **Theory of computation → Sparsification and spanners**; • **Computing methodologies → Machine learning algorithms;**

Additional Key Words and Phrases: Spectral graph theory, spectral graph sparsification, directed graphs, laplacian solver, PageRank

ACM Reference Format:
Ying Zhang, Zhiqiang Zhao, and Zhuo Feng. 2024. diGRASS: Directed Graph Spectral Sparsification via Spectrum-Preserving Symmetrization. *ACM Trans. Knowl. Discov. Data.* 18, 4, Article 102 (February 2024), 25 pages. https://doi.org/10.1145/3639568

1 INTRODUCTION

Graph-based analysis is an essential technique that has been widely adopted in many **electronic design automation** (**EDA**) problems, such as the tasks for logic synthesis and verification, layout optimization, **static timing analysis** (**STA**), network partitioning/decomposition, circuit modeling and simulation, and so on. In recent years, several research problems for simplifying large graphs leveraging spectral graph theory have been extensively studied by mathematics and **theoretical computer science** (**TCS**) researchers [5, 12, 13, 23, 26, 35, 40]. Recent *spectral graph sparsification* research allows constructing nearly-linear-sized subgraphs that can well preserve the spectral (structural) properties of the original graph, such as the the first few eigenvalues and eigenvectors of the graph Laplacian. The related results can potentially lead to

This work is supported in part by the National Science Foundation under Grants CCF-2041519 (CAREER), CCF-2021309 (SHF), and CCF-2011412 (SHF)..
Y. Zhang and Z. Zhao contributed equally to this research.
Authors' address: Y. Zhang, Z. Zhao, and Z. Feng, Stevens Institute of Technology, 1 Castle Point Terrace, Hoboken, NJ 07030, USA; e-mails: {yzhan232, zzhao76, zfeng12}@stevens.edu.

ACM 1556-4681/2024/02-ART102
https://doi.org/10.1145/3639568

the development of a variety of *nearly-linear time* numerical and graph algorithms for solving large sparse matrices and **partial differential equations** (**PDEs**), graph-based **semi-supervised learning** (**SSL**), computing the stationary distributions of Markov chains and personalized PageRank vectors, spectral graph partitioning and data clustering, max flow and multi-commodity flow of undirected graphs, nearly-linear time circuit simulation and verification algorithms, and so on [10, 12, 13, 18, 22, 24, 40, 41, 43].

However, there is not a unified approach that allows for truly-scalable spectral sparsification of directed graphs. For example, the state-of-the-art sampling-based methods for spectral sparsification are only applicable to undirected graphs [24, 38, 41]; the latest theoretical breakthrough in spectral sparsification of directed graphs [11] can only handle strongly-connected directed graphs,[1] which inevitably limits its applications when confronting real-world graphs, since many directed graphs may not be strongly connected, such as the graphs used in chip design automation (e.g., timing analysis) tasks as well as the graphs used in machine learning and data mining tasks.

Consequently, there is still a pressing need for the development of highly-robust (theoretically-rigorous) and truly-scalable (nearly-linear complexity) algorithms for reducing real-world large-scale directed graphs while preserving key graph spectral (structural) properties. In summary, we make the following contributions:

(1) We prove the existence of nearly-linear-sized spectral sparsifiers for directed graphs whose symmetrized undirected graphs only contain non-negative edge weights and introduce a practically-efficient yet unified spectral sparsification approach (diGRASS) that allows simplifying real-world, large-scale (un)directed graphs with guaranteed preservation of the original graph spectra.
(2) We show that leveraging a scalable spectral matrix perturbation analysis for constructing ultra-sparse subgraphs will allow us to well preserve the key eigenvalues and eigenvectors of the original directed graph Laplacians.
(3) Our approach is applicable to a much broader range of directed graphs when comparing with the state-of-the-arts that may only be applicable to specific types of graphs, such as undirected or strongly-connected directed graphs.
(4) Through extensive experiments for real-world directed graphs, diGRASS has been leveraged for computing PageRank vectors, spectral partitioning of directed graphs, and solving directed graph Laplacian matrices.

The spectrally-sparsified directed graphs constructed by diGRASS will potentially lead to the development of much faster numerical and graph-related algorithms. For example, spectrally-sparsified social (data) networks allow for more efficient modeling and analysis of large social (data) networks; spectrally-sparsified neural networks allow for more scalable model training and processing in emerging machine learning tasks; spectrally-sparsified web-graphs allow for much faster computations of personalized PageRank vectors; spectrally-sparsified integrated circuit networks will lead to more efficient partitioning, modeling, simulation, optimization and verification of large chip designs, and so on.

The rest of this article is organized as follows. Section 2 describes recent works related to spectral algorithms for directed graphs and the key idea of the proposed method. Section 3 introduces the background of the (un)directed graphs and spectral graph sparsification. Section 4 introduces a novel theoretical framework for unified spectral sparsification of directed graphs. Section 5 introduces a practically-efficient algorithm for spectral directed graph sparsification. Section 6 describes

[1] A strongly connected directed graph is a directed graph in which any node can be reached from any other node along with direction.

several important applications of the proposed diGRASS algorithm. Section 7 demonstrates comprehensive experiment results of diGRASS for a variety of real-world, large-scale directed graphs, which is followed by the conclusion of this work in Section 8.

2 RELATED WORKS

This section firstly provides a simple overview of undirected graph sparsification. Then we introuduce the existing directed graph symmetrization methods that convert directed graphs into undirected ones such that existing sparsification algorithms on undirected graphs can be directly utilized. At last, prior theoretical directed graph sparsification algorithms are introduced briefly.

2.1 Graph Sparsification

Graph sparsification aims at finding a subgraph (sparsifier) that has the same set of vertices but much fewer edges than the original graph. There are several types of graph sparsifiers proposed for undirected graphs. Graph spanners [2–4, 34] are trying to preserve the pair-wise shortest-path distance between the original graph and the sparsifier. Cut sparsifiers [7, 21] are targeting preserving the cut values between cuts. Spectral sparsification methods preserve the graph spectral (structural) properties, such as distances between vertices, effective resistances, cuts in the graph, as well as the stationary distributions of Markov chains [12, 13, 40]. Therefore, spectral graph sparsification is a much stronger notion than cut sparsification, and more spectral related sparsification methods are proposed in recent years, such as spectral preservation of pseudoinverse for the graph Laplacian [29] and linear-sized sparsifier [28].

2.2 Directed Graph Symmetrization

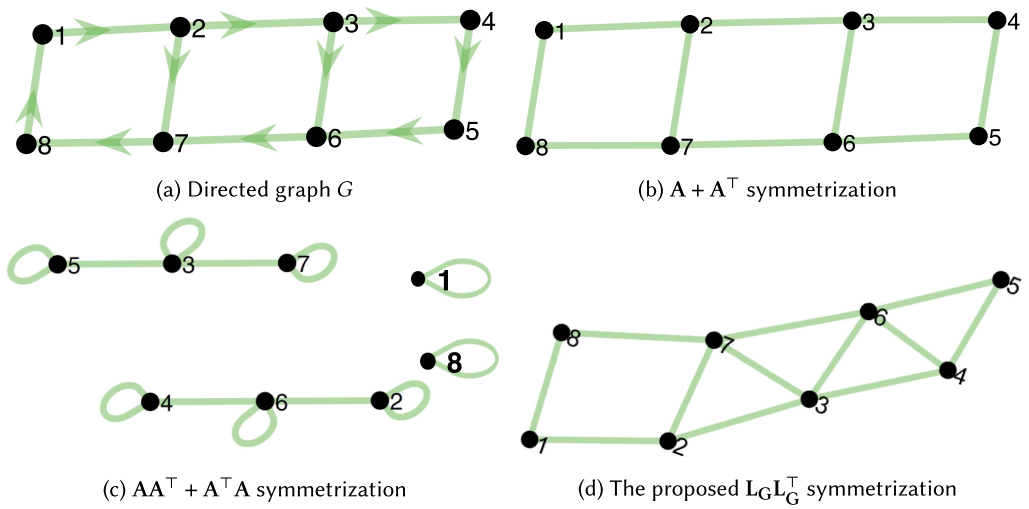
When dealing with directed graphs, it's natural to convert directed graphs into undirected ones so that existing undirected graph algorithms can be subsequently leveraged. The related transforming procedures are called symmetrization methods. We will review three existing graph symmetrization methods, including the $A + A^\top$ [2] symmetrization, bibliometric symmetrization methods, and the random-walk symmetrization.

— $\mathbf{A} + \mathbf{A}^\top$ **symmetrization** simply ignores the edges' directions, which is the simplest and most efficient way for directed graph symmetrization. However, edge directions may play an important role in directed graphs. As shown in Figure 1, edges $(8, 1)$ and $(4, 5)$ seem to have the equal importance in the symmetrized undirected graph $\mathbf{A} + \mathbf{A}^\top$. However, in the original directed graph, edge $(8, 1)$ is much more important than edge $(4, 5)$, since removing edge $(8, 1)$ will lead to the loss of more connections in the directed graph. For example, removing edge $(4, 5)$ will only affect the walks from node 4 to any other nodes as well as walks from any other nodes to node 5. However, if we remove edge $(8, 1)$ in the directed graph, it will affect the walks from node 8 to any other nodes and the walks to node 1; there will also be no access from nodes 5, 6, 7, and 8 to nodes 1, 2, 3, and 4.

— **Bibliographic symmetrization** [37] adopts $\mathbf{A}\mathbf{A}^\top + \mathbf{A}^\top\mathbf{A}$ as the adjacency matrix after symmetrization to take the in-going and out-going edges into consideration. However, it cannot be scaled to large-scale graphs since it will create much denser undirected graphs after symmetrization. Also, disconnected graphs can be created due to the $\mathbf{A}\mathbf{A}^\top + \mathbf{A}^\top\mathbf{A}$ symmetrization, as shown in Figure 1.

— **Random-walk symmetrization** [11] is based on random walks and allows normalized cut to be preserved after symmetrization. This is also a new symmetrization approach used in recent work for defining the Laplacian matrix of directed graphs. When defining the Laplacian matrix, we

[2]The definition for the adjacency matrix of (un)directed graphs A is introduced in Section 4.



(a) Directed graph G (b) $\mathbf{A} + \mathbf{A}^\top$ symmetrization

(c) $\mathbf{A}\mathbf{A}^\top + \mathbf{A}^\top\mathbf{A}$ symmetrization (d) The proposed $\mathbf{L}_G\mathbf{L}_G^\top$ symmetrization

Fig. 1. Converting a directed graph G in (a) into undirected ones using $\mathbf{A} + \mathbf{A}^\top$, $\mathbf{A}\mathbf{A}^\top + \mathbf{A}^\top\mathbf{A}$, and the proposed $\mathbf{L}_G\mathbf{L}_G^\top$ as shown in Figures (b)-(d), respectively.

can apply the random walk for aperiodic graphs, or lazy random walk scheme for periodic graphs. In [11], Cheeger's inequality has been extended to directed graphs and plays a significant role in spectral analysis of directed graphs. It connects Cheeger constant (conductance) with spectral properties (eigenvalues of the graph Laplacian) of a graph. It also provides the bound for the smallest eigenvalue of the directed graph Laplacian. However, the related theoretical results can only be applied to strongly-connected aperiodic directed graphs, which are rare to find in real-world applications.

2.3 Directed Graph Sparsification Algorithms

Refs. [13] and [12] expanded the scope and work on not only strongly-connected graphs but also Eulerian graphs. However, there are obvious limitations with this approach. For example, a random graph needs to be converted into an Eulerian graph via an Eulerian scaling procedure by introducing additional edges, changing the directions of the edges, or reweighing the edges, which may jeopardize the original graphs' spectral properties [13]. In addition, the Eulerian scaling is very timing consuming for large-scale graphs. Lastly, even though the complexity of algorithms is nearly linear-time, it is still not fast in practice for different applications, such as solving asymmetric linear systems, computing the stationary distribution of a Markov chain or computing expected commute time in a directed graph, and so on.

In [9], the authors design cut sparsifiers and sketches for directed graphs. Cut is a property that connects between undirected and directed graphs. How to construct cut sparsifiers for directed graphs really depends on cut balance, which is the ratio between incoming and outgoing edges in any given cut.

3 BACKGROUND

3.1 Definitions and Preliminaries

Undirected graph. Consider a weighted, undirected graph $G = (V, E, \omega)$ with $n = |V|$ and $m = |E|$, where V denotes a set of vertices, n denotes the number of vertices, E denotes a set of edges, m denotes the number of edges, and ω denotes a weight function that assigns a positive weight to

each edge. The adjacency matrix of graph G can be defined as follows:

$$A_G(p,q) = \begin{cases} \omega(p,q) & \text{if } (p,q) \in E \\ 0 & \text{if otherwise .} \end{cases} \tag{1}$$

The Laplacian matrix can be computed by

$$\mathbf{L_G} = \mathbf{D_G} - \mathbf{A_G}, \tag{2}$$

where $\mathbf{D_G}$ is a diagonal matrix with elements $D_G(p,p) = \sum_{t \neq p} \omega(p,t)$. For any real vector $\mathbf{x} \in \mathbb{R}^n$, the Laplacian quadratic form of graph G is defined as $\mathbf{x}^\top \mathbf{L_G} \mathbf{x} = \sum_{(p,q)\in E} \omega(p,q)(x(p) - x(q))^2$. Recall that the undirected graph Laplacian is defined in Equation (2). Alternatively, the undirected graph Laplacian can also be written as [38]

$$\mathbf{L_G} = \mathbf{B}^\top \mathbf{W} \mathbf{B}, \tag{3}$$

where matrix $\mathbf{W}$ is the diagonal matrix with $W(p,p)$ to be the node degree for node p and matrix $\mathbf{B}$ is shown as

$$B(p,v) = \begin{cases} 1 & \text{if } v \text{ is } p\text{th edge's head;} \\ -1 & \text{if } v \text{ is } p\text{th edge's tail;} \\ 0 & \text{otherwise .} \end{cases} \tag{4}$$

Directed graph. Consider a directed graph $G = (V, E_G, w_G)$ with V denoting the set of vertices, E_G representing the set of directed edges, and w_G denoting the associated edge weights. Let $n = |V|$, $m = |E_G|$ be the size of node and edge set. In the following, we denote the diagonal matrix by $\mathbf{D_G}$ with $D_G(i,i)$ being equal to the (weighted) outdegree of node i, as well as the adjacency matrix of G by $\mathbf{A_G}$:

$$A_G(i,j) = \begin{cases} w_G(i,j) & \text{if } (i,j) \in E_G \\ 0 & \text{otherwise .} \end{cases} \tag{5}$$

Then the directed Laplacian matrix can be constructed as follows [13, 14]:

$$\mathbf{L_G} = \mathbf{D_G} - \mathbf{A_G^\top}. \tag{6}$$

The directed graph Laplacian matrix can also be constructed as $\mathbf{L_G} = \mathbf{B}^\top \mathbf{W} \mathbf{C}$, where $\mathbf{W}$ and $\mathbf{B}$ are defined the same as above, while matrix $\mathbf{C}$ is a signed edge-vertex incidence (injection) matrix defined as follows:

$$C(p,v) = \begin{cases} 1 & \text{if } v \text{ is } p\text{th edge's head;} \\ 0 & \text{if } v \text{ is } p\text{th edge's tail;} \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

For better illustration, we have summarized the most-frequently used symbols in our article in Table 1. It can be shown that any directed (undirected) graph Laplacian constructed using Equation (6) will satisfy the following properties: **(I)** Each column (and row) sum is equal to zero; **(II)** All off-diagonal elements are non-positive; **(III)** The Laplacian matrix is asymmetric (symmetric) and indefinite (positive semidefinite).

3.2 Spectral Graph Sparsification

Spectral sparsifier was first introduced by Spielman and Teng [40]. Given an undirected graph with n vertices and m edges, a nearly-linear time algorithm was introduced for building $(1 \pm \epsilon)$ spectral sparsifiers with $O(n \log n/\epsilon^2)$ edges in [38]. S is said to be a $(1 \pm \epsilon)$ spectral sparsifier of G if the following inequality holds for any $x \in \mathbb{R}^n$:

$$(1-\epsilon) x^\top L_G x \leq x^\top L_S x \leq (1+\epsilon) x^\top L_G x, \tag{8}$$

Table 1. Summary of Symbols Used in This Article

Before symmetrization	After symmetrization
$G = (V, E_G, w_G)$: (un)directed graph	$G_u = (V, E_{G_u}, w_{G_u})$: undirected graph
$S = (V, E_S, w_S)$: sparsifier of G	$S_u = (V, E_{S_u}, w_{S_u})$: sparsifier of G_u
V: node set	V: node set
$n = \|V\|$: number of nodes	$n = \|V\|$: number of nodes
E_G: edge set	E_{G_u}: edge set
$m_G = \|E_G\|$: number of edges in E_G	$m_{G_u} = \|E_{G_u}\|$: number of edges in G_u
E_S: edge set of its sparsifier	E_{S_u}: edge set of the symmetrization's sparsifier
$m_S = \|E_S\|$: number of edges in E_S	$m_{S_u} = \|E_{S_u}\|$: number of edges in E_{S_u}
$\mathbf{L_G}$: Laplacian matrix of G	$\mathbf{L_{G_u}}$: Laplacian matrix of G_u
$\mathbf{L_S}$: Laplacian matrix of sparsifier S	$\mathbf{L_{S_u}}$: Laplacian matrix of sparsifier S_u

where $\mathbf{L_G}$ and $\mathbf{L_S}$ denote the **symmetric diagonally dominant** (**SDD**) Laplacian matrices of graphs G and S, respectively. The key to the analysis of spectral sparsifier S, which is the improved construction of Equation (8), is to observe the following equation [6]:

$$\frac{\mathbf{x}^\top \mathbf{L_S} \mathbf{x}}{\sigma} \leq \mathbf{x}^\top \mathbf{L_G} \mathbf{x} \leq \sigma \mathbf{x}^\top \mathbf{L_S} \mathbf{x}, \tag{9}$$

where $\mathbf{x} \in \mathbb{R}^n$. Relative condition number can be defined as $\kappa(\mathbf{L_G}, \mathbf{L_S}) \leq \sigma^2$, implying that a smaller relative condition number or σ^2 corresponds to a higher (better) spectral similarity between two graphs.

4 A THEORETICAL FRAMEWORK FOR UNIFIED SPECTRAL SPARSIFICATION

In this section, we provide an innovative method to convert a directed graph into an undirected one with the proposed $\mathbf{L_G L_G^\top}$ symmetrization. We will also introduce the spectral properties of the new symmetrization scheme, as well as the proof for the existence of nearly-linear-sized spectral sparsifier for a directed graph under certain conditions.

4.1 Our Contribution: The $\mathbf{L_G L_G^\top}$ Symmetrization Scheme [44]

For directed graphs, the subgraph S can be considered spectrally similar to the original graph G if the condition number or the ratio between the largest and smallest singular values of $\mathbf{L_S^+ L_G}$ is close to 1, where $\mathbf{L_S^+}$ denotes the Moore–Penrose pseudoinverse of $\mathbf{L_S}$. Spectral sparsification of directed graphs is equivalent to finding an ultra-sparse subgraph S such that the condition number of $(\mathbf{L_S^+ L_G})^\top (\mathbf{L_S^+ L_G})$ is small enough. Note that the singular values of $\mathbf{L_S^+ L_G}$ are the square roots of eigenvalues of $(\mathbf{L_S^+ L_G})^\top (\mathbf{L_S^+ L_G})$, and $(\mathbf{L_S^+ L_G})^\top (\mathbf{L_S^+ L_G})$ can be written into $\mathbf{L_G^\top (L_S L_S^\top)^+ L_G}$. Although $\mathbf{L_G^\top (L_S L_S^\top)^+ L_G}$ is not equal to $(\mathbf{L_S L_S^\top})^+ (\mathbf{L_G L_G^\top})$, they do share the same eigenvalues under special conditions according to the following theorem [20]:

Theorem 1. *Consider matrices* $\mathbf{X} \in \mathbb{R}^{m',n'}$ *and* $\mathbf{Y} \in \mathbb{R}^{n',m'}$ *with* $m' \leq n'$. *Then the* n' *eigenvalues of* $\mathbf{YX}$ *are the* m' *eigenvalues of* $\mathbf{XY}$ *together with* $n' - m'$ *zeroes; that is* $p_{\mathbf{YX}}(t) = t^{n'-m'} p_{\mathbf{XY}}(t)$. *If* $m' = n'$ *and at least one of* $\mathbf{X}$ *or* $\mathbf{Y}$ *is nonsingular, then* $\mathbf{XY}$ *and* $\mathbf{YX}$ *are similar.*

Based on Therorem 1, $\mathbf{L_G^\top (L_S L_S^\top)^+ L_G}$ and $(\mathbf{L_S L_S^\top})^+ (\mathbf{L_G L_G^\top})$ will share the same eigenvalues. Under this condition, spectral sparsification of directed graphs is equivalent to finding an ultra-sparse subgraph S such that the condition number of $(\mathbf{L_S L_S^\top})^+ (\mathbf{L_G L_G^\top})$ is small enough. Theorem 2 shows both $\mathbf{L_G L_G^\top}$ and $\mathbf{L_S L_S^\top}$ are the Laplacian matrices of some undirected graphs.

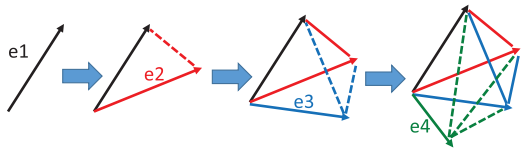


Fig. 2. Edge coupling during directed Laplacian symmetrization.

Theorem 2. *For any (un)directed graph $G = (V, E_G, w_G)$ and its Laplacian $\mathbf{L_G}$, its symmetrized undirected graph $G_u = (V, E_{G_u}, w_{G_u})$ can be obtained via Laplacian symmetrization $\mathbf{L_{G_u}} = \mathbf{L_G L_G^\top}$, where $\mathbf{L_{G_u}}$ is **positive semi-definite (PSD)** and will have the all-one vector as its null space.*

Proof. The row sum of the Laplacian matrix equals to zero, which can be proved as follows:

$$\begin{aligned} & L_{G_u}(i,i) + \sum_{j,j\neq i} L_{G_u}(i,j) \\ &= \sum_k L_G(i,k)L_G(i,k) + \sum_{j,j\neq i}\sum_k L_G(j,k)L_G(i,k) \\ &= \sum_k L_G(i,k)\left(L_G(i,k) + \sum_{j,j\neq i} L_G(j,k)\right) = 0, \end{aligned} \tag{10}$$

which indicates the all-one vector is the subspace of the null space of $\mathbf{L_{G_u}}$. Meanwhile, the column sum of $\mathbf{L_{G_u}}$ equals to zero, which can be proved by the same way. Also, it's very straightforward to prove $\mathbf{L_{G_u}}$ is PSD since $\mathbf{x}^\top \mathbf{L_{G_u}} \mathbf{x} = \mathbf{x}^\top \mathbf{L_G L_G^\top} \mathbf{x} = \|\mathbf{L_G^\top x}\| \geq 0$ holds for any real vector $\mathbf{x} \in \mathbb{R}^{|V|}$.

It can be shown that G_u will contain negative edge weights under the following condition:

$$\begin{aligned} & \sum_k (A_G(k,i)D_G(k,j) + D_G(i,k)A_G(j,k)) > \\ & \sum_k A_G(k,i)A_G(k,j). \end{aligned} \tag{11}$$

The edges will be coupled together and cause a denser graph in $\mathbf{L_{G_u}}$, if a node has more than one outgoing edge. As an example shown in Figure 2, when edge $e2$ is added into the initial graph G that includes a single edge $e1$, an extra edge (shown in red dashed line) coupling with $e1$ will be created in the resultant undirected graph G_u; similarly, when an edge $e3$ is further added, two extra edges coupling with $e1$ and $e2$ will be created in G_u. When the last edge $e4$ is added, It forms a clique. □

4.2 Why not $\mathbf{L_G^\top L_G}$ Symmetrization?

$\mathbf{L_G L_G^\top}$ symmetrization is a novel spectrum-preserving Laplacian symmetrization procedure for converting directed graphs into undirected ones. On the other hand, $\mathbf{L_G^\top L_G}$ does not work for this purpose since $\mathbf{L_G^\top L_G}$ does not correspond to the Laplacian of an undirected graph. Since the row sum of $\mathbf{L_G}$ is not zero, the row sum of the $\mathbf{L_G^\top L_G}$ will not be zero shown as follows:

$$\begin{aligned} & (\mathbf{L_G^\top L_G})(i,i) + \sum_{j,j\neq i} (\mathbf{L_G^\top L_G})(i,j) \\ &= \sum_k \mathbf{L_G}(k,i)\mathbf{L_G}(k,i) + \sum_{j,j\neq i}\sum_k \mathbf{L_G}(k,i)\mathbf{L_G}(k,j) \\ &= \sum_k \mathbf{L_G}(k,i)\left(\mathbf{L_G}(k,i) + \sum_{j,j\neq i} \mathbf{L_G}(k,j)\right) \neq 0. \end{aligned} \tag{12}$$

Although $\mathbf{L}_G^\top \mathbf{L}_G$ is a PSD matrix, the all-one vector is not its null space, and existing methods for spectral sparsification of undirected graphs [5, 43] cannot be exploited for sparsifying directed graphs.

4.3 Existence of Nearly-linear-sized Spectral Sparsifier

In this section, we prove the existence of nearly-linear-sized spectral sparsifier for directed graphs under the condition that their corresponding undirected graphs (obtained through the proposed Laplacian symmetrization scheme) only contain non-negative edge weights.

Lemma 3. *Let $\epsilon > 0$, and $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m$ denote a set of vectors in $\mathbb{R}^n$ that allow expressing the identity decomposition as [5]*

$$\sum_{1 \le i \le m} \mathbf{u}_i \mathbf{u}_i^\top = \mathbf{id}_{\mathbb{R}^n}, \tag{13}$$

where $\mathbf{id}_{\mathbb{R}^n} \in \mathbb{R}^{n \times n}$ denotes the identity matrix. Then there exists a series of non-negative coefficients $\{t_i\}_{i=1}^m$ such that $|\{t_i | t_i \neq 0\}| = O(n/\epsilon^2)$, and

$$\begin{aligned} (1-\epsilon)\mathbf{x}^\top \mathbf{id}_{\mathbb{R}^n}\mathbf{x} \quad &\le \quad \sum_i t_i \mathbf{x}^\top \mathbf{u}_i \mathbf{u}_i^\top \mathbf{x} \\ &\le \quad (1+\epsilon)\mathbf{x}^\top \mathbf{id}_{\mathbb{R}^n}\mathbf{x}. \quad \forall \mathbf{x} \in \mathbb{R}^n. \end{aligned} \tag{14}$$

Theorem 4. *For an undirected graph $G_u = (V, E_{G_u}, w_{G_u})$ converted from a directed graph $G = (V, E_G, w_G)$ via Laplacian symmetrization $\mathbf{L}_{G_u} = \mathbf{L}_G \mathbf{L}_G^\top$, there exists a $(1+\epsilon)$-spectral sparsifier $S_u = (V, E_{S_u}, w_{S_u})$ that can be constructed with $O(n/\epsilon^2)$ PSDs such that the corresponding undirected graph Laplacian $\mathbf{L}_{S_u} = \mathbf{L}_S \mathbf{L}_S^\top$ satisfies the following condition for any $\mathbf{x} \in \mathbb{R}^{|V|}$ [27]:*

$$(1-\epsilon)\mathbf{x}^\top \mathbf{L}_{G_u}\mathbf{x} \le \mathbf{x}^\top \mathbf{L}_{S_u}\mathbf{x} \le (1+\epsilon)\mathbf{x}^\top \mathbf{L}_{G_u}\mathbf{x}. \tag{15}$$

Proof. Lemma (3) proves the existence of the sparsifier for an undirected graph with non-negative edge weights. Given a directed graph G with m_G edges, it can be shown that the Laplacian of the symmetrized undirected graph G_u can be expressed as combination of m_G PSD matrices rather than m_{G_u} PSD matrices. The key of our approach is to construct a set of vectors $\mathbf{u}_1, \ldots, \mathbf{u}_{m_G}$ in $\mathbb{R}^{|V|}$ such that $\mathbf{u}_i$ can be expressed as an identity decomposition shown in Equation (13). Since $\mathbf{L}_{G_u}$ and its Moore–Penrose inverse (pseudoinverse) $\mathbf{L}_{G_u}^+$ can be written as

$$\mathbf{L}_{G_u} = \sum_{j=1}^{n-1} \lambda_j' \mathbf{u}_j' \mathbf{u}_j'^\top, \quad \mathbf{L}_{G_u}^+ = \sum_{j=1}^{n-1} \frac{1}{\lambda_j'} \mathbf{u}_j' \mathbf{u}_j'^\top, \tag{16}$$

where vectors $\mathbf{u}_i'$ and λ_i' are the eigenvector and eigenvalue, respectively, it can be shown that

$$\mathbf{L}_{G_u}\mathbf{L}_{G_u}^+ = \sum_{j=1}^{n-1} \mathbf{u}_j' \mathbf{u}_j'^\top = \mathbf{id}_{\mathbf{L}_{G_u}}, \tag{17}$$

where $\mathbf{id}_{\mathbf{L}_{G_u}}$ is the identity on $\mathbf{im}(\mathbf{L}_{G_u}) = \mathbf{ker}(\mathbf{L}_{G_u})^\top$. In the following, we show how to construct vectors $\mathbf{u}_i$ for $i = 1, \ldots, m_G$. The undirected Laplacian after symmetrization can be written as $\mathbf{L}_{G_u} = \mathbf{B}^\top \mathbf{W}_o \mathbf{B}$ with $\mathbf{W}_o = \mathbf{W}\mathbf{C}\mathbf{C}^\top\mathbf{W}$. Consequently, $\mathbf{U}_{n \times m_G}$ matrix with $\mathbf{u}_i$ for $i = 1, \ldots, m_G$ as its column vectors can be constructed as

$$\mathbf{U}_{n \times m_G} = [\mathbf{u}_1, \ldots, \mathbf{u}_{m_G}] = \mathbf{L}_{G_u}^{+/2} \mathbf{B}^\top \mathbf{W}_o^{1/2}. \tag{18}$$

It can be shown that $\mathbf{U}_{\mathbf{n}\times\mathbf{m_G}}$ will satisfy the following equation:

$$\begin{aligned}\mathbf{U}_{\mathbf{n}\times\mathbf{m_G}}\mathbf{U}_{\mathbf{n}\times\mathbf{m_G}}^{\top} &= \sum_{\mathbf{i}=\mathbf{1}}^{\mathbf{m_G}} \mathbf{u_i}\mathbf{u_i}^{\top} = {\mathbf{L_{G_u}}}^{+/2}\mathbf{B}^{\top}\mathbf{W_o}\mathbf{B}{\mathbf{L_{G_u}}}^{+\top/2} \\ &= {\mathbf{L_{G_u}}}^{+/2}\mathbf{L_{G_u}}{\mathbf{L_{G_u}}}^{+\top/2} = \mathbf{id}_{\mathbf{L_{G_u}}}\end{aligned}. \tag{19}$$

According to Lemma 3, we can always construct a diagonal matrix $\mathbf{T} \in \mathbb{R}^{\mathbf{m}\times\mathbf{m}}$ with t_i as its ith diagonal element. Then there will be at most $O(n/\epsilon^2)$ positive diagonal elements in $\mathbf{T}$, which allows constructing $\mathbf{L_{S_u}}$ as

$$\mathbf{L_{S_u}} = \mathbf{B}^{\top}\mathbf{W_o}^{1/2}\mathbf{T}\mathbf{W_o}^{1/2}\mathbf{B} \tag{20}$$

that corresponds to the directed subgraph S for achieving $(1+\epsilon)$-spectral approximation of G as required by Equation (15).

Since matrix $\mathbf{W_o}$ is a symmetric positive semidefinite matrix and $\mathbf{T}$ is a symmetric diagonal matrix, $\mathbf{L_{S_u}}$ can be further written into

$$\mathbf{L_{S_u}} = \mathbf{B}^{\top}\mathbf{W_o}^{\frac{1}{2}}\mathbf{T}^{\frac{1}{2}}\mathbf{T}^{\frac{1}{2}}\mathbf{W_o}^{\frac{1}{2}}\mathbf{B} = \left(\mathbf{B}^{\top}\mathbf{W_o}^{\frac{1}{2}}\mathbf{T}^{\frac{1}{2}}\right)\left(\mathbf{B}^{\top}\mathbf{W_o}^{\frac{1}{2}}\mathbf{T}^{\frac{1}{2}}\right)^{\top}. \tag{21}$$

Therefore, the Laplacian of the directed graph $\mathbf{L_S}$ can be expressed as

$$\mathbf{L_S} = \mathbf{B}^{\top}\mathbf{W_o}^{\frac{1}{2}}\mathbf{T}^{\frac{1}{2}}. \tag{22}$$

Also, the following inequality holds for any $\mathbf{x} \in \mathbb{R}^{|V|}$

$$(\mathbf{1}-\epsilon)\mathbf{x}^{\top}\mathbf{id}_{\mathbf{L_{G_u}}}\mathbf{x} \le \sum_{\mathbf{i}} \mathbf{t_i}\mathbf{u_i}\mathbf{u_i}^{\top} \le (\mathbf{1}+\epsilon)\mathbf{x}^{\top}\mathbf{id}_{\mathbf{L_{G_u}}}\mathbf{x}. \tag{23}$$

Since $\sum_i \mathbf{t_i}\mathbf{u_i}\mathbf{u_i}^{\top} = \mathbf{U}\mathbf{T}\mathbf{U}^{\top}$ and we have

$$\begin{aligned}\mathbf{U}\mathbf{T}\mathbf{U}^{\top} &= \left(\mathbf{L}_{\mathbf{G_u}}^{+/2}\mathbf{B}^{\top}\mathbf{W_o}^{1/2}\right)\mathbf{T}\left(\mathbf{L}_{\mathbf{G_u}}^{+/2}\mathbf{B}^{\top}\mathbf{W_o}^{1/2}\right)^{\top} \\ &= \mathbf{L}_{\mathbf{G_u}}^{+/2}\mathbf{L}_{\mathbf{S_u}}\mathbf{L}_{\mathbf{G_u}}^{+/2},\end{aligned} \tag{24}$$

while Equation (23) can be proved through the following steps based on the Courant–Fischer Theorem:

$$\begin{aligned}&1-\epsilon \le \frac{\mathbf{y}^{\top}\mathbf{U}\mathbf{T}\mathbf{U}^{\top}\mathbf{y}}{\mathbf{y}^{\top}\mathbf{y}} \le 1+\epsilon \quad \forall \mathbf{y} \in im(\mathbf{L_{G_u}}) \\ \Longleftrightarrow\ &1-\epsilon \le \frac{\mathbf{y}^{\top}\mathbf{L}_{\mathbf{G_u}}^{+/2}\mathbf{L_{S_u}}\mathbf{L}_{\mathbf{G_u}}^{+/2}\mathbf{y}}{\mathbf{y}^{\top}\mathbf{y}} \le 1+\epsilon \quad \forall \mathbf{y} \in im(\mathbf{L_{G_u}}) \\ \Longleftrightarrow\ &1-\epsilon \le \frac{\mathbf{x}^{\top}\mathbf{L}_{\mathbf{G_u}}^{\frac{1}{2}}\mathbf{L}_{\mathbf{G_u}}^{+/2}\mathbf{L_{S_u}}\mathbf{L}_{\mathbf{G_u}}^{+/2}\mathbf{L}_{\mathbf{G_u}}^{\frac{1}{2}}\mathbf{x}}{\mathbf{x}^{\top}\mathbf{L}_{\mathbf{G_u}}^{\frac{1}{2}}\mathbf{L}_{\mathbf{G_u}}^{\frac{1}{2}}\mathbf{x}} \le 1+\epsilon \quad \forall \mathbf{x} \perp \mathbf{1} \\ \Longleftrightarrow\ &1-\epsilon \le \frac{\mathbf{x}^{\top}\mathbf{L_{S_u}}\mathbf{x}}{\mathbf{x}^{\top}\mathbf{L_{G_u}}\mathbf{x}} \le 1+\epsilon \quad \forall \mathbf{x} \perp \mathbf{1} \\ \Longleftrightarrow\ &(1-\epsilon)\mathbf{x}^{\top}\mathbf{L_{G_u}}\mathbf{x} \le \mathbf{x}^{\top}\mathbf{L_{S_u}}\mathbf{x} \le (1+\epsilon)\mathbf{x}^{\top}\mathbf{L_{G_u}}\mathbf{x} \quad \forall \mathbf{x} \perp \mathbf{1}.\end{aligned} \tag{25}$$

Equation (25) demonstrates that S_u is a $(1+\epsilon)$-spectral sparsifier of graph G_u under specific conditions. □

Theorem 4 proves that there exists an undirected graph S_u and the connection between graph S_u and the original directed graph G. The next key step is to show that $\mathbf{L}_{\mathbf{S}_u}$ can be factorized into the product of a directed graph Laplacian $\mathbf{L}_\mathbf{S}$ and its transpose. Also, Theorem 4 does not immediately imply a sparse structure in $\mathbf{L}_\mathbf{S}$. To prove the existence of nearly-linear-sized spectral sparsifier for a directed graph, we further assume the undirected graph G_u obtained through the proposed Laplacian symmetrization only contains edges with non-negative weights. Next, the following Lemma 5 can be exploited to prove the existence of nearly-linear-sized $\mathbf{L}_\mathbf{S}$ based on Equation (22):

Lemma 5. *[SparseCholesky Algorithm [25]] Given an $n \times n$ undirected Laplacian matrix $\mathbf{L_u}$ with $O(m)$ non-positive off-diagonal elements (non-negative edge weights), the SparseCholesky Algorithm [25] runs in expected time $O(m \log^3 n)$ and computes a permutation Π, a lower triangular matrix $\mathcal{L}$ with $O(m \log^3 n)$ nonzero entries, and a diagonal matrix $\mathbf{D}$ such that with probability $1 - \frac{1}{poly(n)}$, we have*

$$\frac{1}{2}\mathbf{x}^\top \mathbf{L_u}\mathbf{x} \leq \mathbf{x}^\top \mathbf{Z}\mathbf{x} \leq \frac{3}{2}\mathbf{x}^\top \mathbf{L_u}\mathbf{x}, \tag{26}$$

where $\mathbf{Z} = \mathbf{\Pi}\mathcal{L}\mathbf{D}\mathcal{L}^\top\mathbf{\Pi}^\top$, and $\mathbf{Z}$ has a sparse Cholesky factorization.

Reference [25] provides a nearly-linear time algorithm for constructing $\mathcal{L}$ using the sparsified Cholesky factorization method, which is a process of constructing clique structure of the Schur complement. More importantly, [25] demonstrates that $\mathcal{L}$ is a lower diagonal matrix which always corresponds to a feed-forward directed graph. As a result, we can conclude that given a Laplacian matrix of an undirected graph with non-negative edge weights, it can be always factorized into the product of a nearly-linear-sized directed Laplacian matrix and its transpose through the LDL^T decomposition (sparsified Cholesky factorization).

Combining Theorem 4 and Lemma 5 will allow us to prove the following main theorem.

Theorem 6. *For any given directed graph $G = (V, E_G, w_G)$, when its undirected graph $G_u = (V, E_{G_u}, w_{G_u})$ obtained via the proposed Laplacian symmetrization only contains non-negative edge weights, there exists a $(1+\epsilon)$-spectral sparsifier $S = (V, E_S, w_S)$ with $O(n \log^3 n/\epsilon^2)$ edges.*

However, the above theorem has its own limitations. For example, when comparing with the works of [11, 13, 14] which can preserve the cut in the sparsifier, ours cannot.

5 DIGRASS: A PRACTICALLY-EFFICIENT ALGORITHM FOR SPECTRAL SPARSIFICATION OF DIRECTED GRAPHS

To apply our theoretical results to deal with real-world directed graphs, the following concerns should be addressed in advance:

- The undirected graph $\mathbf{L_G}\mathbf{L_G^\top}$ may become too dense to compute and thus may impose high cost during spectral sparsification.
  As we introduced in Section 4.1, the $\mathbf{L_G}\mathbf{L_G^\top}$ symmetrization scheme will create extra edges if a node has more than one outgoing edge in $\mathbf{L_G}$, as shown in Figure 2. Under this condition, it may be possible to directly perform symmetrization on $\mathbf{L_G}$ if $\mathbf{L_G}$ is relatively sparse. However, for general cases where $\mathbf{L_G}$ may be very dense, the generated $\mathbf{L}_{\mathbf{G_u}}$ will be much denser due to the edge coupling effect, which will inevitably impose high computational and memory cost for following spectral sparsification procedure. To achieve a general algorithmic framework for handling directed graph with various densities, it is necessary to solve this issue during the framework design.
- It can be quite challenging to convert the sparsified undirected graph to its corresponding directed sparsifier $\mathbf{L_S}$, even when $\mathbf{L}_{\mathbf{S_u}}$ is available.

There is no guarantee that the $\mathbf{L}_S$ is an one-to-one correspondence to $\mathbf{L}_{S_u}$. For example, it is possible that multiple $\mathbf{L}_S$ correspond to the same symmetrized undirected graph Laplacian $\mathbf{L}_{S_u}$. So it can be challenging to convert the $\mathbf{L}_{S_u}$ back to $\mathbf{L}_S$, even when $\mathbf{L}_{S_u}$ is available. While the coupling edges generated during symmetrization will make the situation even harder.

To address the above concerns for unified spectral graph sparsification, we propose a practically-efficient framework with following desired features: **(1)** our approach does not require to explicitly compute $\mathbf{L}_G\mathbf{L}_G^\top$ but only the matrix-vector multiplications; **(2)** our approach can effectively identify the most spectrally-critical edges for dramatically decreasing the relative condition number; **(3)** although our approach requires to compute $\mathbf{L}_S\mathbf{L}_S^\top$, the $\mathbf{L}_{S_u}$ matrix density can be effectively controlled by carefully pruning spectrally-similar edges through the proposed edge similarity checking scheme.

5.1 Initial Sparsifier Construction

Motivated by the recent research on low-stretch spanning trees [1, 17] and spectral perturbation analysis [18, 43] for nearly-linear-time spectral sparsification of undirected graphs, we propose a practically-efficient algorithm for sparsifying general directed graphs by first constructing the initial subgraph sparsifiers of directed graphs through the following steps:

- **Step 1:** Compute $\mathbf{D}^{-1}(\mathbf{A}_G + \mathbf{A}_G^\top)$ as a new adjacency matrix, where $\mathbf{D}$ denotes the diagonal matrix with each element equal to the row (column) sum of $(\mathbf{A}_G + \mathbf{A}_G^\top)$. Recent research shows such split transformations can effectively reduce graph irregularity while preserving critical graph connectivity, distance between node pairs, the minimal edge weight in the path, as well as outdegrees and indegrees when using push-based and pull-based vertex-centric programming [33].
- **Step 2:** Construct a **maximum spanning tree** (**MST**) based on $\mathbf{D}^{-1}(\mathbf{A}_G + \mathbf{A}_G^\top)$, which allows effectively controlling the number of outgoing edges for each node so that the resultant undirected graph after symmetrization will not be too dense.
- **Step 3:** Recover the direction of each edge in the MST and make sure each node of its sparsifier has at least one outgoing edge if there are more than one in the original graph for achieving stronger connectivity in the initial directed sparsifier.

5.2 Spectral Sensitivity of Off-subgraph Edges

As aforementioned, when the condition number of $\mathbf{L}_{S_u}^+\mathbf{L}_{G_u}$ is small, the condition number of $\mathbf{L}_S^+\mathbf{L}_G$ will be small, which represents that graph S is a good spectral sparsifier for graph G. To this end, we will exploit the following spectral perturbation analysis framework for computing spectral sensitivity of each off-subgraph edges. For the generalized eigenvalue problem

$$\mathbf{L}_{G_u}\mathbf{v_i} = \lambda_i\mathbf{L}_{S_u}\mathbf{v_i}, \quad \text{for } i = 1, \ldots, n \tag{27}$$

let matrix $\mathbf{V} = [\mathbf{v_1}, \ldots, \mathbf{v_n}]$. Then $\mathbf{v_i}$ and λ_i can be constructed to satisfy the following orthogonality requirement:

$$\mathbf{v_i}^\top\mathbf{L}_{G_u}\mathbf{v_j} = \begin{cases}\lambda_i, & i = j\\ 0, & i \neq j\end{cases} \quad \text{and} \quad \mathbf{v_i}^\top\mathbf{L}_{S_u}\mathbf{v_j} = \begin{cases}1, & i = j\\ 0, & i \neq j.\end{cases} \tag{28}$$

Consider the following first-order generalized eigenvalue perturbation problem:

$$\mathbf{L}_{G_u}(\mathbf{v_i} + \delta\mathbf{v_i}) = (\lambda_i + \delta\lambda_i)(\mathbf{L}_{S_u} + \delta\mathbf{L}_{S_u})(\mathbf{v_i} + \delta\mathbf{v_i}), \tag{29}$$

where a small perturbation $\delta \mathbf{L}_{\mathbf{S_u}}$ in $\mathbf{L}_{\mathbf{S_u}}$ is introduced, leading to the perturbed generalized eigenvalues and eigenvectors $\lambda_i + \delta\lambda_i$ and $\mathbf{v_i} + \delta\mathbf{v_i}$. By only keeping the first-order terms, Equation (29) becomes

$$\mathbf{L}_{\mathbf{G_u}}\delta\mathbf{v_i} = \lambda_\mathbf{i}\mathbf{L}_{\mathbf{S_u}}\delta\mathbf{v_i} + \lambda_\mathbf{i}\delta\mathbf{L}_{\mathbf{S_u}}\mathbf{v_i} + \delta\lambda_\mathbf{i}\mathbf{L}_{\mathbf{S_u}}\mathbf{v_i}. \tag{30}$$

Let $\delta\mathbf{v_i} = \sum_\mathbf{j} \psi_{\mathbf{i,j}}\mathbf{v_j}$, then Equation (30) can be expressed as

$$\sum_\mathbf{j} \psi_{\mathbf{i,j}}\mathbf{L}_{\mathbf{G_u}}\mathbf{v_j} = \lambda_\mathbf{i}\mathbf{L}_{\mathbf{S_u}}\left(\sum_\mathbf{j} \psi_{\mathbf{i,j}}\mathbf{v_j}\right) + \lambda_\mathbf{i}\delta\mathbf{L}_{\mathbf{S_u}}\mathbf{v_i} + \delta\lambda_\mathbf{i}\mathbf{L}_{\mathbf{S_u}}\mathbf{v_i}. \tag{31}$$

Based on the orthogonality properties in Equation (28), multiplying $\mathbf{v_i}$ to both sides of Equation (31) results in

$$\lambda_\mathbf{i}\delta\mathbf{L}_{\mathbf{S_u}}\mathbf{v_i} + \delta\lambda_\mathbf{i}\mathbf{L}_{\mathbf{S_u}}\mathbf{v_i} = \mathbf{0}, \tag{32}$$

which further leads to

$$\frac{\delta\lambda_i}{\lambda_i} = -\mathbf{v_i}^\top \delta\mathbf{L}_{\mathbf{S_u}}\mathbf{v_i}. \tag{33}$$

Then the task of spectral sparsification of general (un)directed graphs will require to recover as few as possible off-subgraph edges to the initial directed subgraph S such that the largest eigenvalues, or the condition number of $\mathbf{L}_{\mathbf{S_u}}^{+}\mathbf{L}_{\mathbf{G_u}}$ can be dramatically reduced. Expand $\delta\mathbf{L}_{\mathbf{S_u}}$ with only the first-order terms as

$$\delta\mathbf{L}_{\mathbf{S_u}} = \delta\mathbf{L}_\mathbf{S}\mathbf{L}_\mathbf{S}^\top + \mathbf{L}_\mathbf{S}\delta\mathbf{L}_\mathbf{S}^\top, \tag{34}$$

where $\delta\mathbf{L}_\mathbf{S} = w_G(p,q)\mathbf{e}_{\mathbf{p,q}}\mathbf{e}_\mathbf{p}{}^\top$ for $(p,q) \in E_G \setminus E_S$, $\mathbf{e_p} \in \mathbb{R}^n$ denotes the vector with only the p-th element being 1 and others being 0, and $\mathbf{e_{p,q}} = \mathbf{e_p} - \mathbf{e_q}$. The **spectral sensitivity** for each off-subgraph edge (p,q) can be expressed as

$$\zeta_{p,q} = \mathbf{v_i}^\top\left(\delta\mathbf{L}_\mathbf{S}\mathbf{L}_\mathbf{S}^\top + \mathbf{L}_\mathbf{S}\delta\mathbf{L}_\mathbf{S}^\top\right)\mathbf{v_i}. \tag{35}$$

It is obvious that Equation (35) can be leveraged to rank the spectral importance of each off-subgraph edge. Consequently, spectral sparsification of general graphs can be achieved by only recovering a few dissimilar off-subgraph edges with large spectral sensitivity values. In this work, the following method based on t-step power iterations is proposed for efficient computation of dominant generalized eigenvectors

$$\mathbf{v_1} \approx \mathbf{h_t} = \left(\mathbf{L}_{\mathbf{S_u}}^{+}\mathbf{L}_{\mathbf{G_u}}\right)^\mathbf{t}\mathbf{h_0}, \tag{36}$$

where $\mathbf{h_0}$ denotes a random vector. When the number of power iterations is small (e.g., $t \leq 3$), $\mathbf{h_t}$ will be a linear combination of the first few dominant generalized eigenvectors corresponding to the largest few eigenvalues. Then the spectral sensitivity for the off-subgraph edge (p,q) can be approximately computed by

$$\zeta_{p,q} \approx \mathbf{h_t}^\top\left(\delta\mathbf{L}_\mathbf{S}\mathbf{L}_\mathbf{S}^\top + \mathbf{L}_\mathbf{S}\delta\mathbf{L}_\mathbf{S}^\top\right)\mathbf{h_t}. \tag{37}$$

The computation of $\mathbf{h_t}$ through power iterations requires solving the linear system of equations $\mathbf{L}_{\mathbf{S_u}}\mathbf{x} = \mathbf{b}$ for t times. Note that only $\mathbf{L}_{\mathbf{S_u}}$ needs to be explicitly computed for generalized power iterations. The **Lean Algebraic Multigrid** (**LAMG**) [30] solver is leveraged for computing $\mathbf{h_t}$, which can handle undirected graphs with negative edge weights and has an empirical $O(|E_{S_u}|)$ complexity for solving Laplacian matrices $\mathbf{L}_{\mathbf{S_u}}$.

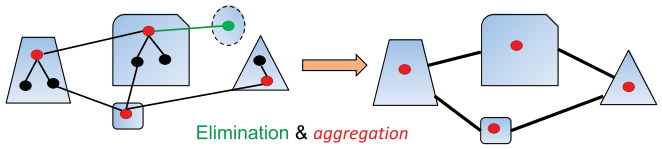


Fig. 3. LAMG setup phase.

5.3 Lean Algebraic Multigrid (LAMG)

The setup phase of LAMG contains two main steps [30], as shown in Figure 3. First, a nodal elimination procedure is performed to eliminate disconnected and low-degree nodes. Next, a node aggregation procedure is applied for aggregating strongly connected nodes according to the following affinity metric c_{uv} for nodes u and v:

$$c_{uv} = \frac{(\mathbf{X}(\mathbf{u},:),\mathbf{X}(\mathbf{v},:))^2}{(\mathbf{X}(\mathbf{u},:),\mathbf{X}(\mathbf{u},:))(\mathbf{X}(\mathbf{v},:),\mathbf{X}(\mathbf{v},:))} \quad \text{with } (\mathbf{x},\mathbf{y}) = \Sigma_{k=1}^{K} x(k)\cdot y(k). \tag{38}$$

where $\mathbf{X} = (\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(\mathbf{K})})$ is computed by applying a few **Gauss–Seidel** (**GS**) relaxations using K initial random vectors to the linear system equation $\mathbf{L}_{\mathbf{S_u}}\mathbf{x} = 0$. Let $\tilde{\mathbf{x}}$ represent the approximation of the true solution $\mathbf{x}$ after applying several GS relaxations to $\mathbf{L}_{\mathbf{S_u}}\mathbf{x} = 0$. Due to the smoothing property of GS relaxation, the latest error can be expressed as $\mathbf{e_s} = \mathbf{x} - \tilde{\mathbf{x}}$, which will only contain the smooth components of the initial error, while the highly oscillating modes will be effectively damped out [8]. Nodes u and v are considered strongly connected to each other if $\mathbf{X}(\mathbf{u},:)$ and $\mathbf{X}(\mathbf{v},:)$ are highly correlated for all the K test vectors (or a larger c_{uv} value), which thus should be aggregated to form a coarse level node.

Once the multilevel hierarchical representations of the original graph (Laplacians) have been created, **algebraic multigrid** (**AMG**) solvers can be built and subsequently leveraged to solve large Laplacian matrices efficiently.

5.4 Edge Spectral Similarities

The proposed spectral sparsification algorithm will first sort all off-subgraph edges according to their spectral sensitivities in descending order $(p_1,q_1),(p_2,q_2),\ldots$ and then select top few off-subgraph edges to be recovered to the initial subgraph. To avoid recovering redundant edges into the subgraph, it is indispensable to check the edge similarities: only the edges that are not similar to each other will be added to the initial sparsifier. To this end, we exploit the following spectral embedding scheme for distinguishing off-subgraph edges leveraging approximate dominant generalized eigenvectors $\mathbf{h_t}$ computed by Equation (36):

$$\psi_{p,q}(h_t) = \sum_k w_{p,q_k}\mathbf{h_t}^{\top}\left(\mathbf{e_{p,q}}\mathbf{e}_{\mathbf{p,q_k}}^{\top} + \mathbf{e_{p,q_k}}\mathbf{e}_{\mathbf{p,q}}^{\top}\right)\mathbf{h_t}, \tag{39}$$

where (p,q_k) are the directed edges sharing the same head with (p,q) but different tails. Then the proposed scheme for checking the spectral similarity of two off-subgraph edges will include the following steps:

Step 1: Perform t-step power iterations with $r = O(\log n)$ initial random vectors $\mathbf{h}_0^{(1)},\ldots,\mathbf{h}_0^{(r)}$ to compute r approximate dominant generalized eigenvectors $\mathbf{h}_t^{(1)},\ldots,\mathbf{h}_t^{(r)}$;
Step 2: For each edge (p,q), compute an r-dimensional spectral embedding vector $\mathbf{s_{p,q}} \in \mathbb{R}^r$ with $s_{p,q}(r) = \psi_{p,q}(h_t^{(r)})$;

Step 3: Check the similarity of two off-subgraph edges (p_i, q_i) and (p_j, q_j) with

$$\text{SpectralSim}(i,j) = 1 - \frac{||\mathbf{s}_{\mathbf{p_i},\mathbf{q_i}} - \mathbf{s}_{\mathbf{p_j},\mathbf{q_j}}||}{\max(||\mathbf{s}_{\mathbf{p_i},\mathbf{q_i}}||, ||\mathbf{s}_{\mathbf{p_j},\mathbf{q_j}}||)}. \tag{40}$$

If $\text{SpectralSim}(i,j) < \varrho$ for a given threshold ϱ, edge (p_i, q_i) is considered spectrally dissimilar to (p_j, q_j). Given a list of candidate off-subgraph edges, Algorithm 2 is proposed for edge similarity checking.

5.5 Algorithm Flow and Complexity of diGRASS

Algorithm 1 shows the algorithm flow for directed graph sparsification, where $\mathbf{L_G}$ is the Laplacian matrix for original graph, $\mathbf{L_S}$ is the Laplacian matrix of initial spanning tree, d_{out} is the user-defined outgoing degree for nodes, and λ_{limit} is the desired maximum generalized eigenvalue. Algorithm 2

ALGORITHM 1: The diGRASS Algorithm Flow

Input: $\mathbf{L_G}$, $\mathbf{L_S}$, d_{out}, λ_{limit}, α, ϱ
Output: $\mathbf{L_S}$

Compute the dominant generalized eigenvector $\mathbf{h_t}$, and its eigenvalue $\lambda_{\max}$;
while $\lambda_{\max} > \lambda_{\text{limit}}$ **do**
  Compute the spectral sensitivity $\zeta_{p,q}$ of each off-subgraph edge $(p,q) \in E_{G\backslash S}$;
  Sort edge spectral sensitivities in descending order and include the top $\alpha\%$ off-subgraph edges into the candidate edge list $\tilde{E}_{\text{list}} = [(p_1, q_1), (p_2, q_2), \ldots]$;
  Form the final edge list E_{list} that only includes spectrally-dissimilar off-subgraph edges obtained by using Edge_Similarities_Checking($\tilde{E}_{\text{list}}, \mathbf{L_G}, \mathbf{L_S}, d_{\text{out}}, \varrho$);
  Update $\tilde{S} = S + E_{\text{list}}$ and compute the latest dominant generalized eigenvector $\tilde{\mathbf{h}}_\mathbf{t}$, and its eigenvalue $\tilde{\lambda}_{\max}$ based on $\mathbf{L_G}$ and $\mathbf{L}_{\tilde{\mathbf{S}}}$;
  if $\tilde{\lambda}_{\max} < \lambda_{\max}$ **then**
    Update $S = \tilde{S}$, $\mathbf{h_t} = \tilde{\mathbf{h}}_\mathbf{t}$, $\lambda_{\max} = \tilde{\lambda}_{\max}$;
  end if
end while
Return $\mathbf{L_S}$.

ALGORITHM 2: Edge_Similarities_Checking

Input: $\tilde{E}_{\text{list}}$, $\mathbf{L_G}$, $\mathbf{L_S}$, d_{out}, ϱ
Output: E_{list}

Perform t-step power iterations with $r = O(\log n)$ initial random vectors $\mathbf{h_0^{(1)}}, \ldots, \mathbf{h_0^{(r)}}$ to compute r approximate dominant generalized eigenvectors $\mathbf{h_t^{(1)}}, \ldots, \mathbf{h_t^{(r)}}$;
Compute a r-dimensional embedding vector $\mathbf{s}_{p_i,q_i} \in \mathbb{R}^r$ for $\forall (p_i, q_i) \in \tilde{E}_{\text{list}}$;
let $E_{\text{list}} = [(p_1, q_1)]$;
for i=2:$|\tilde{E}_{\text{list}}|$ **do**
  Calculate the spectral similarity score $\text{SpectralSim}(i,j)$ between (p_i, q_i) and every edge (p_j, q_j) in $\tilde{E}_{\text{list}}$;
  if $\text{SpectralSim}(i,j) < \varrho$ and $d_{p_j}, d_{q_j} < d_{\text{out}}$ for $\forall (p_j, q_j) \in \tilde{E}_{\text{list}}$ **then**
    $E_{\text{list}} = [\tilde{E}_{\text{list}}; (p_i, q_i)]$;
  end if
end for
Return E_{list} ;

obtains the edges after checking edges similarities for the off-subgraph edges, where $\tilde{E}_{\text{list}}$ is the set of off-subgraph edges; E_{list} is the set of edges that will be added into sparsifier; d_p is the outgoing degree for node p. The complexity has been summarized as follows:

- **(a)** Generate an initial subgraph S from the original directed graph in $O(m \log n)$ or $O(m+n \log n)$ time;
- **(b)** Compute the approximate dominant eigenvector $\mathbf{h_t}$ and the spectral sensitivity of each off-subgraph edge in $O(m)$ time;
- **(c)** Recover a small amount of spectrally-dissimilar off-subgraph edges into the latest subgraph S according to their spectral sensitivities and similarities in $O(m)$ time;
- **(d)** Repeat steps **(b)** and **(c)** until the desired condition number or spectral similarity is achieved.

6 APPLICATIONS OF DIRECTED GRAPH SPARSIFICATION

6.1 Directed Laplacian Solver

Recent research has focused on developing more efficient algorithms for solving undirected Laplacians [22, 24]. In this work, we will focus on solving asymmetric Laplacian matrices that correspond to directed graphs: solving the following linear system equations $\mathbf{L_G x = b}$, where the **right-hand-side** (**RHS**) vector $\mathbf{b}$ lies in the left singular vector space, will be equivalent to solving the following problem:

$$\mathbf{L_G L_G^\top L_G^{\top +} x = b}. \tag{41}$$

Let $\mathbf{y = L_G^{\top +} x}$, then we will first solve $\mathbf{L_{G_u} y = b}$. Once $\mathbf{y}$ is obtained, we can get the solution $\mathbf{x = L_G^\top y}$. Since $\mathbf{L_{G_u}}$ is a much denser matrix, $\mathbf{L_{G_u}}$ should not be explicitly formed when solving $\mathbf{L_{G_u} y = b}$. To this end, iterative methods such as the **preconditioned conjugate gradient** (**PCG**) method can be leveraged for solving $\mathbf{L_{G_u} y = b}$ with $\mathbf{L_{S_u}}$ as the preconditioner. Note that only $\mathbf{L_{S_u}}$ will be explicitly computed during PCG iterations.

The directed graph sparsifier can also be directly leveraged as a preconditioner for solving $\mathbf{L_G x = b}$ using existing iterative methods, such as the **generalized minimal residual** (**GMRES**) method [36]. GMRES is a widely-adopted Krylov-subspace iterative method for solving asymmetric matrices. Given an initial solution vector $\mathbf{x_0}$, GMRES gradually improves the solution $\mathbf{x_m}$ of the mth iteration by minimizing the residue as follows:

$$\mathbf{x_m} = \underset{\mathbf{z} \in \mathbf{x_0} + \mathcal{K}_m(\mathbf{L_G}, \mathbf{r_0})}{\operatorname{argmin}} \|\mathbf{L_G z - b}\|_2, \tag{42}$$

where $\mathbf{r_0 = b - L_G x_0}$, and $\mathcal{K}_m(\mathbf{L_G}, \mathbf{r_0}) = span\{\mathbf{r_0}, \mathbf{L_G r_0}, \mathbf{L_G^2 r_0}, \ldots, \mathbf{L_G^{m-1} r_0}\}$ denotes the Krylov subspace.

6.2 (Personalized) PageRank Vectors

The idea of PageRank is to give a measurement of the importance for each web page. For example, PageRank algorithm aims at finding the most popular web pages, while the personalized PageRank algorithm aims at finding the pages that users will most likely visit. To state it mathematically, the PageRank vector π satisfies the following equation:

$$\pi = (c\mathbf{A_G^\top D_G}^{-1} + (1-c)\mathbf{v_0 1}^\top)\pi, \tag{43}$$

where $0 < c < 1$ is the damping constant, and vector $\mathbf{v_0}$ with non-negative coordinates, satisfying $\mathbf{1}^\top \mathbf{v_0} = \mathbf{1}$, is the personalization vector. The original, non-personalized definition of the PageRank is described when $\mathbf{v_0} = \frac{1}{n}\mathbf{1}$. Meanwhile, $\mathbf{D_G^{-1}}$ can not be defined if there exist nodes that have no outgoing edges. To deal with such situation, a self-loop with a small edge weight can be added for each node.

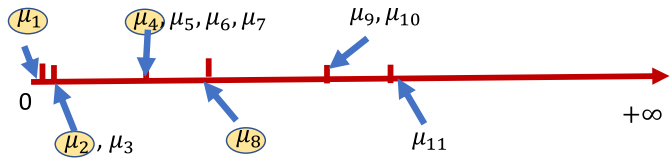


Fig. 4. Eigenvalues of $\mathbf{L}_{G_u}$ for the directed graph in Figure 5 .

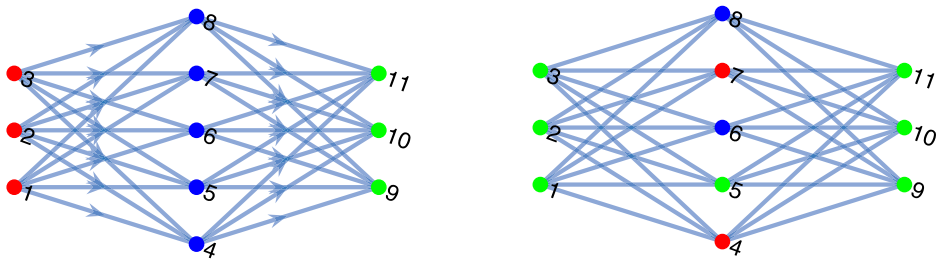


Fig. 5. Spectral partitioning of directed (left) and undirected graphs (right). The nodes within the same cluster are assigned the same color.

6.3 Directed Graph Partitioning

It has been shown that partitioning and clustering of directed graphs can play very significant roles in a variety of applications related to machine learning [31], data mining and circuit synthesis and optimization [32], and so on. However, the efficiency of existing methods for partitioning directed graphs strongly depends on the complexity of the underlying graphs [31]. For an undirected graph, the eigenvectors corresponding to the first few smallest eigenvalues can be utilized for the spectral partitioning purpose [39]. For a directed graph G on the other hand, the eigenvectors corresponding to the first few different smallest eigenvalues of Laplacian $\mathbf{L}_{G_u}$ will be required for directed graph partitioning. The eigenvalues according to the symmetrization of the directed graph in Figure 5 have a few multiplicities, which are shown in Figure 4. The partitioning result of the directed graph in Figure 5 will depend on the eigenvectors that correspond to eigenvalues of $\mu_1, \mu_2, \mu_4, \mu_8$. As shown in Figure 5, the spectral partitioning results can be quite different between the directed and undirected graph with the same set of nodes and edges.

7 EXPERIMENTAL RESULTS

The proposed algorithm for spectral sparsification of directed graphs has been implemented using MATLAB and C++. Extensive experiments have been conducted to evaluate the proposed method with various types of directed graphs obtained from public-domain datasets [15]. To ensure that every node in the graph has at least one out-going edge, we delete the nodes with no out-going edges in the graph.

7.1 Dataset Description

The datasets are from SuiteSparse Matrix Collection [16]. If a node has only incoming edges or is isolated with the rest of nodes, this bode will be removed from the graph. The statistics of datasets are summarized in Table 2. The detailed description for each graph is shown as follows:

— gre_115, gre_185, and gre_1107 are from the Harwell–Boeing collection, which describe the simulation of computer systems.
— hor is from the Harwell–Boeing Collection and it describes a flow network.

Table 2. Statistics of Datasets

Dataset	$\|V\|$	$\|E_G\|$	$\frac{\|E_G\|}{\|V\|}$
gre_115	1.1E2	4.2E2	3.8
gre_185	1.8E2	1.0E3	5.6
gre_1107	1.1E3	5.6E3	5.1
harvard500	0.5E3	2.6E3	5.2
cell1	0.7E4	3.0E4	4.3
hor	0.4E3	3.7E3	9.3
pesa	1.2E4	8.0E4	6.7
big	1.3E3	0.9E5	6.9

Dataset	$\|V\|$	$\|E_G\|$	$\frac{\|E_G\|}{\|V\|}$
wordnet3	7.7E4	1.3E5	1.7
p2p-Gnutella05	3.4E3	1.4E4	4.1
p2p-Gnutella31	1.5E4	5.2E4	3.7
email-Eu-core	1.0E3	2.5E4	25.3
wiki-Vote	7.1E3	1.0E5	14.7
cit-HepTh	2.7E4	3.5E5	13.0

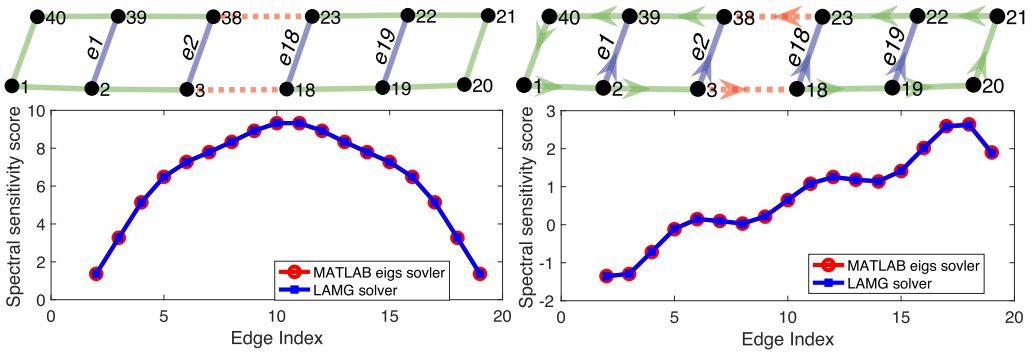


Fig. 6. The spectral sensitivity scores of off-subgraph edges ($e1$ to $e19$ in blue) for the undirected (left) and directed graph (right).

— harvard500 is a web connectivity matrix from Cleve Moler.
— cell1 is a GSM cell traffic matrix from Salvatore Lucifora, Telecom Italia Mobile.
— big and pesa are structure symmetric matrices.
— wordnet3 is a directed multi-relational network.
— p2p-Gnutella31 and p2p-Gnutella05 are Gnutella peer to peer networks.
— email-Eu-core is a relatively denser social network that is generated with e-mail data from a research institute.
— wiki-Vote is a relatively denser social network from the Wikipedia vote dataset.
— cit-HepTh is a high-energy physics theory citation network from arxiv.

7.2 Spectral Edge Sensitivities

Figure 6 shows the spectral sensitivities of all the off-subgraph edges ($e1$ to $e19$ represented with blue color) in both directed and undirected graphs calculated using MATLAB's `eigs` function and the proposed method based on (37) using the LAMG solver, respectively. Meanwhile, the spectral sensitivities of all the off-subgraph edges ($e1$ to $e19$) with respect to the dominant eigenvalues (λ_{max} or λ_1) in both directed and undirected graphs are plotted. We observe that spectral sensitivities for directed and undirected graphs are drastically different from each other. The reason is that the spectral sensitivities for off-subgraph edges in the directed graph depend on the edge directions. It is also observed that the approximate spectral sensitivities calculated by the proposed t-step power iterations with the LAMG solver match the true solution very well for both directed and undirected graphs.

Table 3. Results of Directed Graph Spectral Sparsification

Test Cases	$\|V_G\|$	$\|E_G\|$	$\frac{\|E_{S^0}\|}{\|E_G\|}$	$\frac{\|E_S\|}{\|E_G\|}$	time (s)	$\frac{\lambda_{max,S^0}}{\lambda_{max}}$
gre_115	1.1E2	4.2E2	0.46	0.79	0.05	7.5E3
gre_185	1.8E2	1.0E3	0.25	0.62	0.14	1.1E4
harvard500	0.5E3	2.6E3	0.31	0.40	0.64	1.2E3
cell1	0.7E4	3.0E4	0.31	0.57	3.10	1.0E5
hor	0.4E3	3.7E3	0.23	0.52	0.52	270
pesa	1.2E4	8.0E4	0.27	0.51	8.80	5.3E8
big	1.3E4	0.9E5	0.27	0.49	12.86	4.1E11
gre_1107	1.1E3	5.6E3	0.26	0.39	0.24	1.6E3
wordnet3	7.7E4	1.3E5	0.60	0.85	50.00	223
p2p-Gnutella31	1.5E4	5.2E4	0.33	0.59	11.90	129
p2p-Gnutella05	3.4E3	1.4E4	0.29	0.56	2.64	240
mathworks100	1.0E2	5.5E2	0.20	0.50	0.04	30
email-Eu-core	1.0E3	2.5E4	0.06	0.65	2.03	590
wiki-Vote	7.1E3	1.0E5	0.08	0.54	8.92	3.9E3
cit-HepTh	2.7E4	3.5E5	0.09	0.25	30.30	427

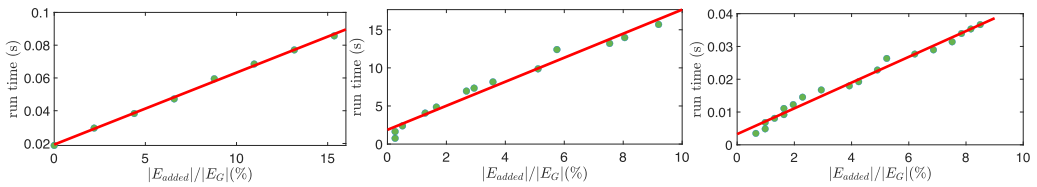


Fig. 7. Runtime scalability for "gre_1107" (left), "big" (middle), "gre_115" (right).

7.3 Directed Graph Sparsification

Table 3 shows comprehensive results on directed graph spectral sparsification for a variety of real-world directed graphs using the proposed method, where $|V_G|(|E_G|)$ denotes the number of nodes (edges) for the original directed graph G; $|E_{S^0}|$ and $|E_S|$ denote the numbers of edges in the initial subgraph S^0 and final spectral sparsifier S. Notice that we will directly apply the MATLAB's `eigs` function if graph size is relatively small ($|E_{S^0}| < 1E4$); otherwise, we will apply LAMG solver for better efficiency when calculating the generalized eigenvector $\mathbf{h_t}$. Note that a small diagonal entry with value of $1e-6$ is added to all symmetrized undirected graphs during the calculation. We report the total runtime for the eigsolver using either the LAMG solver or `eigs` function. $\frac{\lambda_{max,S^0}}{\lambda_{max}}$ denotes the reduction rate of the largest generalized eigenvalue of $\mathbf{L}_{\mathbf{S_u}}^{+}\mathbf{L}_{\mathbf{G_u}}$ from initial sparsifier to final sparsifier.

Ablation study. Since the proposed method is iteratively adding edges for forming the sparsifier. We demonstrate the performance of the runtime and generalized eigenvalue reduction with respect to the number of added edges in the sparsifier. Figure 7 shows the runtime scalability regarding to the number of off-subgraph edges ($|E_{added}|$) added in the final sparsifier for graph "gre_1107" (left), "big" (middle) and "gre_115" (right). It shows that the runtime scales linearly with the added number of edges for all three graphs. Figure 8 shows how $\lambda_{max}(L_{G_u}, L_{S_u})$ is changing when including different number of edges in the sparsifier. We can observe that λ_{max} can be efficiently reduced

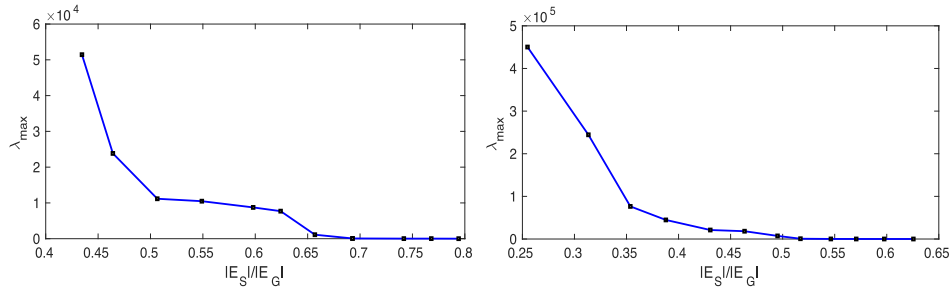


Fig. 8. Eigenvalue change with respect to added number of edges for "gre_115" (left) "gre_185" (right).

Table 4. Comparison of Spectral Sparsification Results

Test cases	GRASS [19]				diGRASS (this work)	
	$\frac{\lvert E_{S'_u}\rvert}{\lvert E_{G'_u}\rvert}$	$\lambda_{max}(\mathbf{L}_{G'_u}, \mathbf{L}_{S'_u})$	$\frac{\lvert E_S\rvert}{\lvert E_G\rvert}$	$\lambda_{max}(\mathbf{L}_{G_u}, \mathbf{L}_{S_u})$	$\frac{\lvert E_S\rvert}{\lvert E_G\rvert}$	$\lambda_{max}(\mathbf{L}_{G_u}, \mathbf{L}_{S_u})$
gre_115	0.92	28	0.44	3760	0.43	**522**
gre_185	0.67	25	0.40	1140	0.41	**170**
gre_1107	0.86	9	0.43	2790	0.43	**147**
harvard500	0.36	13	0.39	5.22E5	0.66	**125**
p2p-Gnutella05	0.55	7	0.55	2.42E5	0.56	**107**
p2p-Gnutella31	0.59	6	0.59	1.4E5	0.59	**224**
big	0.60	7	0.60	8803	0.60	**270**
hor	0.31	17	0.30	209	0.30	**34**
wordnet3	0.78	8	0.79	5.94E4	0.85	**513**

when adding more edges in the sparsifier, especially at the early-stage of sparsifier construction. It also demonstrates that the most spectrally-critical edges can be efficiently identified and included at the early stage comparing to the edges that are less critical.

7.4 Comparison with Prior Method

Since there are no other existing directed graph sparsification methods to be compared, we compare our proposed method with the existing undirected graph sparsification tool GRASS [18, 19, 43]. To this end, we first convert directed graphs into undirected ones (G'_u) using $\mathbf{A} + \mathbf{A}^\top$ symmetrization. Then undirected graph sparsifiers S'_u will be computed by GRASS. In the last, the directed graph sparsifiers can be constructed by recovering edge directions to the undirected sparsifier S'_u. Note that a larger diagonal entry with value of $1e-4$ is added to all symmetrized undirected graphs during the calculation. The experimental results have been shown in Table 4, where λ_{max} represents the largest generalized eigenvalue between the original graph and its final sparsifier. By keeping similar numbers of edges in the sparsifiers, we observe that the proposed spectral sparsification method consistently produces much better spectral sparsifiers than GRASS. Note that for graphs "harvard500" and "wordnet3", we cannot include more edges into the sparsifiers S'_u using GRASS, implying that the final $\lambda_{max}(\mathbf{L}_{G_u}, \mathbf{L}_{S_u})$ cannot be further reduced; on the other hand, our method is able to further reduce its condition number, achieving a much better spectral approximation level.

7.5 Directed Laplacian Solvers

Figure 9 shows the relative residual ($res = \|\mathbf{L}_G\mathbf{x} - \mathbf{b}\|/\|\mathbf{b}\|$) and runtime plots when spectral sparsifiers are applied as the preconditioners for solving the Laplacians of directed graphs "hor",

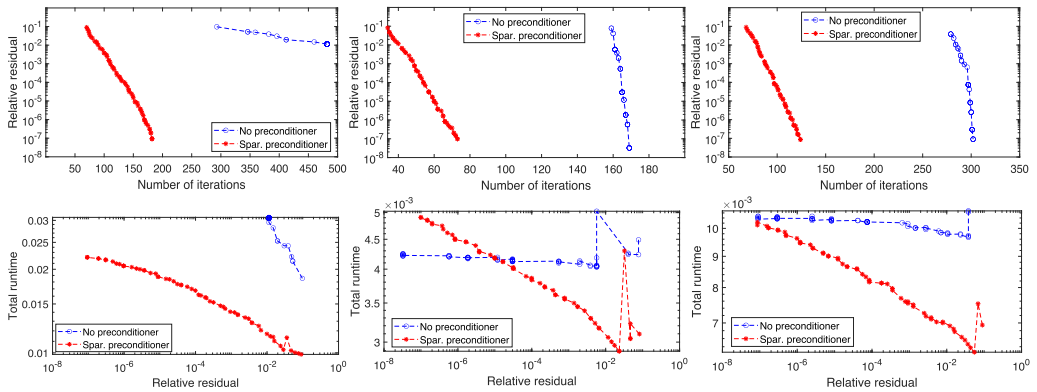


Fig. 9. PCG convergence (the first row) and runtime (the second row) results for graphs "hor", "gre_115" and "gre_185", respectively.

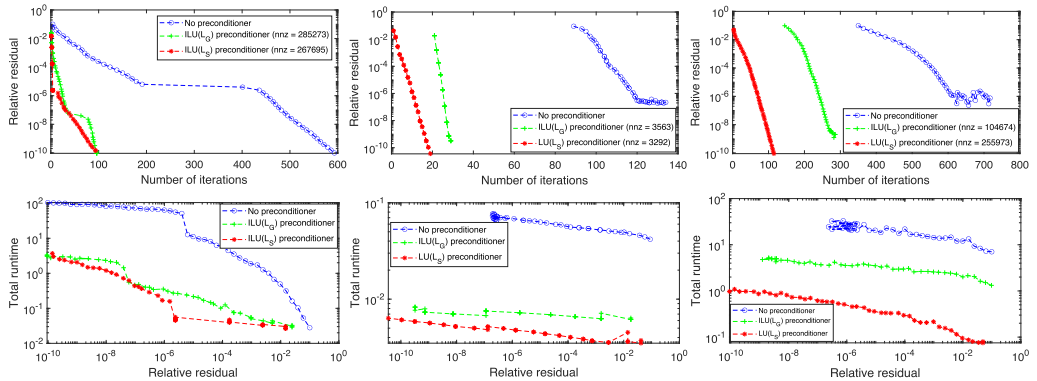
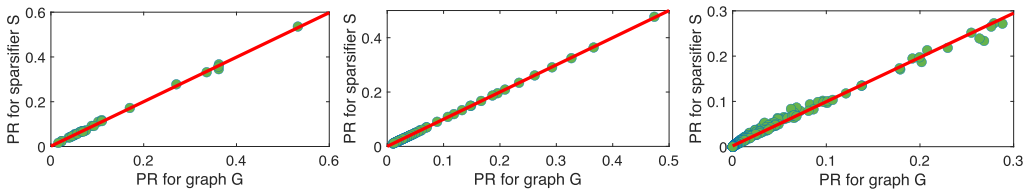


Fig. 10. GMRES convergence (the first row) and runtime (the second row) results for graphs "wordnet3", "harvard" and "big", respectively.

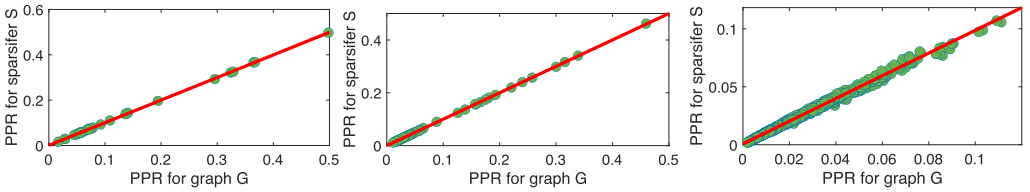
"gre_115" and "gre_185", respectively. As observed, the performance of the PCG solver has been substantially improved by leveraging sparsifier-based preconditioners. Note that for graph "hor" the plain PCG solver without using any preconditioner cannot converge to the desired accuracy within the maximum number of iterations (500 iterations). Figure 10 shows the relative residual and runtime plots when the preconditioners obtained via **Incomplete LU** (**ILU**) factorization of the original directed graphs and their spectral sparsifiers are applied for "wordnet3", "harvard500" and "big", respectively. "ILU(·)" and "LU(·)" indicate that ILU and LU decompositions have been leveraged to construct the preconditioners, respectively. "nnz" denotes the number of nonzeros in the preconditioners. The MATLAB's built-in functions `gmres`, `ilu`, and `lu` with default settings have been applied in our experiments. Note that the GMRES iterations with preconditioners show much faster convergence for all test cases. It is also observed for each test case the preconditioner computed using the directed sparsifier always has lowest number of nonzeros (nnz).

7.6 (Personalized) PageRank Computations

Figure 11 shows the application of the proposed directed graph sparsification for computing (personalized) PageRank vectors with $c = 0.85$, where the correlation of (personalized) PageRank



(a) The correlation of PageRank between itself and its sparsifier for graphs "ibm32" (left), "mathworks100" (middle) and "gre_1107" (right) after smoothing



(b) The correlation of personalized PageRank between itself and its sparsifier for graph "ibm32" (left), "mathworks100" (middle) and "gre_1107" (right) after smoothing

Fig. 11. (Personalized) PageRank Results.

Table 5. Spectral Partitioning Results

Testcase	pesa	gre_115	gre_185	gre_1107	harvard500	hor	big	email-Eu-core
$\frac{\lvert E_S\rvert}{\lvert E_G\rvert}$	0.95	0.79	0.73	0.81	0.66	0.58	0.75	0.62
np	154	5	15	80	14	98	493	120
$np/\lvert V_G\rvert$	0.013	0.043	0.081	0.072	0.028	0.226	0.037	0.121
$cut(G)$	149	148	576	939	9,670	1,037	1,037	29,438
$\theta(G)$	0.068	6.290	12.457	3.688	90.740	12.862	0.314	242.189
$cut(S)$	165	107	308	662	9,748	492	778	12,898
$\theta(S)$	0.072	4.468	6.804	2.439	87.385	6.181	0.236	157.706

results using the original graphs (x-axis) and sparsifiers (y-axis) are plotted for graphs "ibm32" (left), "mathworks100" (middle) and "gre_1107" (right), respectively. Note that a few steps of GS smoothing have been applied to remove the high-frequency errors to obtain the smoothed (personalized) PageRank vectors when using the sparsified graphs. We observe that the (personalized) PageRank vectors obtained from sparsifiers can well approximate the results computed with the original graphs.

7.7 Directed Graph Partitioning

Table 5 shows the detailed partitioning results on different graphs. Since there is no clear clue for spectral directed graph partitioning, we choose to perform spectral partitioning on the symmetrized undirected graph G_u and S_u, where two-way spectral partitioning are applied by utilizing the Fiedler Vector of its Laplacian matrix. np is the number of nodes that share the different partitions when comparing the partitioning results on graph G_u and S_u, where a smaller np indicates a more similar partitioning results between two graphs, thus a better spectral similarity between the original graph and the sparsifier. $np/|V_G|$ can be considered as the percentage of the mismatched node over all node set. cut is the cut value between two partitions, which is equivalent to the number of edges connecting two partitions. θ is the ratio cut [42] value that can be computed with the

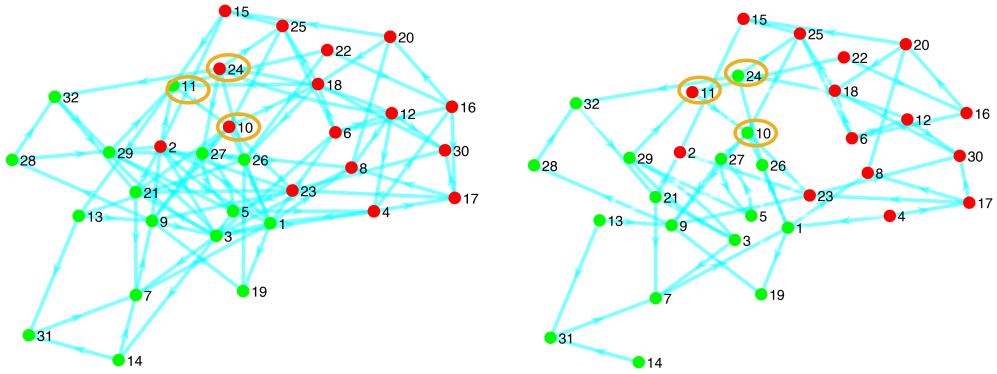


Fig. 12. The partitioning results between G_u (left) and its sparsifier S_u (right) for the "ibm32" graph.

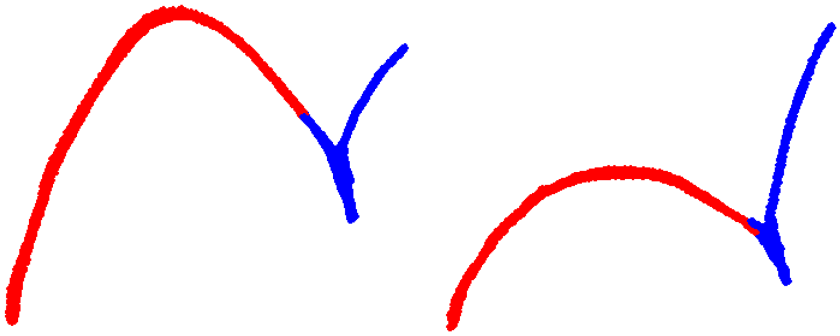


Fig. 13. The partitioning results between G_u (left) and its sparsifier S_u (right) for the "peta" graph.

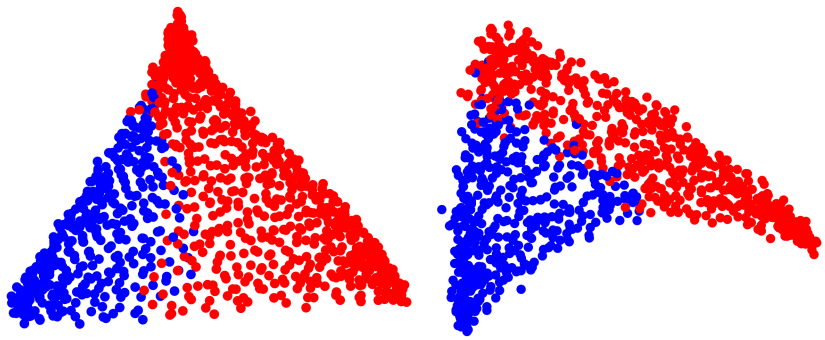


Fig. 14. The partitioning results between G_u (left) and its sparsifier S_u (right) for the "gre_1107" graph.

following equation given the partition V_i and V_j:

$$\theta = \frac{cut(V_i, V_j)}{|V_i|} + \frac{cut(V_i, V_j)}{|V_j|}. \tag{44}$$

Figures 12, 13, 14, and 15 show the partitioning results on the symmetrized graph G_u and its symmetrized sparsifier S_u for "ibm", "peta", "gre_1107", and "big" graphs. As observed, very similar partitioning results have been obtained, indicating well preserved spectral properties within the spectrally-sparsified directed graph.

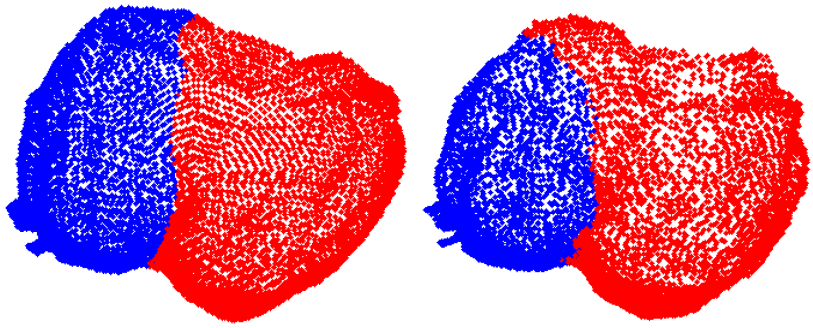


Fig. 15. The partitioning results between G_u (left) and its sparsifier S_u (right) for the "big" graph.

8 CONCLUSIONS

This article proves the existence of nearly-linear-sized spectral sparsifiers for directed graphs under the condition that their corresponding undirected graphs (obtained through the proposed Laplacian symmetrization scheme) only contain non-negative edge weights, and proposes a practically-efficient yet unified spectral graph sparsification framework. Such a novel spectral sparsification approach allows sparsifying real-world, large-scale directed and undirected graphs with guaranteed preservation of the original graph spectral properties. By exploiting a highly-scalable (nearly-linear complexity) spectral matrix perturbation analysis framework for constructing nearly-linear sized (directed) subgraphs, it enables us to well preserve the key eigenvalues and eigenvectors of the original (directed) graph Laplacians. The proposed method has been validated using various kinds of directed graphs obtained from public domain sparse matrix collections, showing promising spectral sparsification results for general directed graphs.

REFERENCES

[1] Ittai Abraham and Ofer Neiman. 2012. Using petal-decompositions to build a low stretch spanning tree. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 395–406.

[2] Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen Kobourov, and Richard Spence. 2020. Graph spanners: A tutorial review. *Computer Science Review* 1, 37 (2020), 100253.

[3] Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. 1993. On sparse spanners of weighted graphs. *Discrete and Computational Geometry* 9, 1 (1993), 81–100.

[4] Surender Baswana and Sandeep Sen. 2007. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures and Algorithms* 30, 4 (2007), 532–563.

[5] Joshua Batson, Daniel Spielman, and Nikhil Srivastava. 2012. Twice-ramanujan sparsifiers. *SIAM Journal on Computing* 41, 6 (2012), 1704–1721.

[6] Joshua Batson, Daniel A. Spielman, Nikhil Srivastava, and Shang-Hua Teng. 2013. Spectral sparsification of graphs: Theory and algorithms. *Communications of the ACM* 56, 8 (2013), 87–94.

[7] András A Benczúr and David R Karger. 1996. Approximating st minimum cuts in Õ (n 2) time. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 47–55.

[8] William L. Briggs, Van Emden Henson, and Steve F. McCormick. 2000. *A Multigrid Tutorial*. Vol. 72. Siam.

[9] Ruoxu Cen, Yu Cheng, Debmalya Panigrahi, and Kevin Sun. 2021. Sparsification of directed graphs via cut balance. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

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

[11] Fan Chung. 2005. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics* 9, 1 (2005), 1–19.

[12] Michael B. Cohen, Jonathan Kelner, Rasmus Kyng, John Peebles, Richard Peng, Anup B. Rao, and Aaron Sidford. 2018. Solving directed laplacian systems in nearly-linear time through sparse LU factorizations. In *Proceedings of the 2018 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 898–909.

[13] Michael B. Cohen, Jonathan Kelner, John Peebles, Richard Peng, Anup B. Rao, Aaron Sidford, and Adrian Vladu. 2017. Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 410–419.

[14] Michael B. Cohen, Jonathan Kelner, John Peebles, Richard Peng, Aaron Sidford, and Adrian Vladu. 2016. Faster algorithms for computing the stationary distribution, simulating random walks, and more. In *Proceedings of the 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 583–592.

[15] T. Davis and Y. Hu. 2011. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software* 38, 1 (2011), 1.

[16] Timothy A Davis and Yifan Hu. 2011. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software* 38, 1 (2011), 1–25.

[17] Michael Elkin, Yuval Emek, Daniel A Spielman, and Shang-Hua Teng. 2008. Lower-stretch spanning trees. *SIAM Journal on Computing* 38, 2 (2008), 608–628.

[18] Zhuo Feng. 2016. Spectral graph sparsification in nearly-linear time leveraging efficient spectral perturbation analysis. In *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 57.

[19] Zhuo Feng. 2020. Grass: Graph spectral sparsification leveraging scalable spectral perturbation analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 12 (2020), 4944–4957.

[20] Roger A. Horn and Charles R. Johnson. 2012. *Matrix Analysis*. Cambridge University Press.

[21] David R. Karger. 1994. Random sampling in cut, flow, and network design problems. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*. 648–657.

[22] Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. 2014. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 217–226.

[23] Pavel Kolev and Kurt Mehlhorn. 2015. Approximate spectral clustering: Efficiency and guarantees. arXiv preprint arXiv:1509.09188.

[24] I. Koutis, G. Miller, and R. Peng. 2010. Approaching optimality for solving SDD linear systems. In *Proceedings of the IEEE FOCS*. 235–244.

[25] Rasmus Kyng and Sushant Sachdeva. 2016. Approximate Gaussian elimination for laplacians-fast, sparse, and simple. In *Proceedings of the 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 573–582.

[26] Yin Tat Lee and He Sun. 2017. An sdp-based algorithm for linear-sized spectral sparsification. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 678–687.

[27] Yin Tat Lee and He Sun. 2017. An sdp-based algorithm for linear-sized spectral sparsification. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 678–687.

[28] Yin Tat Lee and He Sun. 2018. Constructing linear-sized spectral sparsification in almost-linear time. *SIAM Journal on Computing* 47, 6 (2018), 2315–2336.

[29] Huan Li and Aaron Schild. 2018. Spectral subspace sparsification. In *Proceedings of the 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 385–396.

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

[31] Fragkiskos D. Malliaros and Michalis Vazirgiannis. 2013. Clustering and community detection in directed networks: A survey. *Physics Reports* 533, 4 (2013), 95–142.

[32] Giovanni De Micheli. 1994. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill Higher Education.

[33] Amir Hossein Nodehi Sabet, Junqiao Qiu, and Zhijia Zhao. 2018. Tigr: Transforming irregular graphs for GPU-friendly graph processing. In *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 622–636.

[34] David Peleg and Alejandro A Schäffer. 1989. Graph spanners. *Journal of Graph Theory* 13, 1 (1989), 99–116.

[35] Richard Peng, He Sun, and Luca Zanetti. 2015. Partitioning well-clustered graphs: Spectral clustering works. In *Proceedings of the 28th Conference on Learning Theory (COLT)*. 1423–1455.

[36] Youcef Saad and Martin H. Schultz. 1986. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 7, 3 (1986), 856–869.

[37] Venu Satuluri and Srinivasan Parthasarathy. 2011. Symmetrizations for clustering directed graphs. In *Proceedings of the 14th International Conference on Extending Database Technology*. ACM, 343–354.

[38] Daniel Spielman and Nikhil Srivastava. 2011. Graph sparsification by effective resistances. *SIAM Journal on Computing* 40, 6 (2011), 1913–1926.

[39] D. Spielman and Shanghua Teng. 1996. Spectral partitioning works: Planar graphs and finite element meshes. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 96–105.

[40] Daniel Spielman and ShangHua Teng. 2011. Spectral sparsification of graphs. *SIAM Journal on Computing* 40, 4 (2011), 981–1025.

[41] D. Spielman and S. Teng. 2014. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications* 35, 3 (2014), 835–885.
[42] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and Computing* 17, 4 (2007), 395–416.
[43] Z. Feng. 2018. Similarity-aware spectral sparsification by edge filtering. In *Proceedings of the 55th Design Automation Conference (DAC)*. IEEE.
[44] Ying Zhang, Zhiqiang Zhao, and Zhuo Feng. 2019. Towards scalable spectral sparsification of directed graphs. In *Proceedings of the 2019 IEEE International Conference on Embedded Software and Systems (ICESS)*. IEEE, 1–2.

Received 3 March 2022; revised 19 May 2023; accepted 6 December 2023