2020

HIGH-PERFORMANCE SPECTRAL METHODS FOR COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS

Zhiqiang Zhao
Michigan Technological University, qzzhao@mtu.edu

HIGH-PERFORMANCE SPECTRAL METHODS FOR COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS

By

Zhiqiang Zhao

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

In Computer Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2020

This dissertation has been approved in partial fulfillment of the requirements for the Degree of DOCTOR OF PHILOSOPHY in Computer Engineering.

Department of Electrical and Computer Engineering

Dissertation Co-advisor:	*Dr. Zhuo Feng*
Dissertation Co-advisor:	*Dr. Glen E. Archer*
Committee Member:	*Dr. Jiguang Sun*
Committee Member:	*Dr. Chee-Wooi Ten*
Committee Member:	*Dr. Benjamin W. Ong*
Committee Member:	*Dr. Saeid Nooshabadi*
Department Chair:	*Dr. Glen E. Archer*

Dedication

To my family, teachers and friends

who didn't hesitate to support my work at every stage - without which I would neither be who I am nor would this work be what it is today.

Contents

List of Figures

List of Tables

Preface

This dissertation presents my research work during my PhD study in Computer Engineering at Michigan Technological University, which includes previously published papers in Chapter 2 , Chapter 3 and Chapter 4. All the research works presented here were conducted under the supervision of my advisor Dr. Zhuo Feng.

Chapter 2 contains two papers. One paper was published in Proceedings of the 56th Annual Design Automation Conference (DAC), and the second paper will be published in 14th ACM International Conference on Web Searching and Data Mining (WSDM). As the first author of two papers, with the guidance of my advisor, I completed the algorithm design, implementation, and analysis. Papers were completed by my advisor and I. Ying Zhang, as the second author of the second paper, participated in the second paper's manuscript writing.

Chapter 3 contains one paper published in 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). As the first author of the paper, with the guidance of my advisor, I completed the algorithm design, implementation, and analysis. Yongyu Wang, as the second author of the paper, provided me with the Laplacian matrices of MNIST data set. The paper was completed by my advisor and I.

Chapter 4 contains two papers which were published in Proceedings of the 54th Annual Design Automation Conference (DAC) and 2020 Design, Automation & Test in Europe Conference. As the first author of two papers, with the guidance of my advisor, I completed the algorithm design, implementation, and analysis. The two papers were completed by my advisor and I.

Acknowledgments

First of all, I want to give my sincere thanks to my advisor Professor Zhuo Feng, who gave me the full support and guidance throughout my whole PhD studies. He introduced me to the research topics presented in this dissertation. His insight and knowledge has inspired me to sharpen my thinking and elevate my work to a higher level. He has never hesitated to help me on the various of ideas of the research and cares for the growth towards my academic goal.

Next, I would like to thank my co-advisor Professor Glen E. Archer who has been always supportive of my career goals and worked actively to provide me with the opportunities of learning and teaching. It is the training from him that I learned the skills and attitude for both teaching and researching. I am also thankful for his guidance and support for the preparation of my final defense.

Then, I would like to thank all my committee members: Professor Jiguang Sun, Professor Chee-Wooi Ten, Professor Benjamin W. Ong and Professor Saeid Nooshabadi. Thanks for their support and expert advice during my proposal and dissertation process. I am really grateful for all the inspirations and suggestions regarding to the researches from them.

My appreciation also extends to my friends and classmates for their friendship and

support during my whole PhD period. Dr. Xueqian Zhao and Dr. Lengfei Han, who were senior students during my first two years, were always willing to share their valuable experiences on researches with me and helped me a lot on understanding of the related researches, and I am also thankful that they shared their codes with me. Dr. Caoyang Jiang was always willing and enthusiastic to assist in any way he could. And I am grateful for his guidance during my study of programming languages. Collaboration with Dr. Zhaoxiang Jin was a very pleasant experience and I appreciate his patience and inspirations during our collaboration.

Finally, I want to give my full thanks to my family. It is their support and love that motivated me to pursue my dreams and goals, and it is their strength I can rely on when I was frustrated. I want to thank my wife for her full support and encouragements during my PhD study. Her confidence and optimistic attitudes always influence me to think positively, and her sharp insights also inspire me a lot in my life.

Abstract

Recent research shows that by leveraging the key spectral properties of eigenvalues and eigenvectors of graph Laplacians, more efficient algorithms can be developed for tackling many graph-related computing tasks. In this dissertation, spectral methods are utilized for achieving faster algorithms in the applications of very-large-scale integration (VLSI) computer-aided design (CAD).

First, a scalable algorithmic framework is proposed for effective-resistance preserving spectral reduction of large undirected graphs. The proposed method allows computing much smaller graphs while preserving the key spectral (structural) properties of the original graph. Our framework is built upon the following three key components: a spectrum-preserving node aggregation and reduction scheme, a spectral graph sparsification framework with iterative edge weight scaling, as well as effective-resistance preserving post-scaling and iterative solution refinement schemes. We show that the resultant spectrally-reduced graphs can robustly preserve the first few nontrivial eigenvalues and eigenvectors of the original graph Laplacian and thus allow for developing highly-scalable spectral graph partitioning and circuit simulation algorithms.

Based on the framework of the spectral graph reduction, a Sparsified graph-theoretic Algebraic Multigrid (SAMG) is proposed for solving large Symmetric Diagonally

Dominant (SDD) matrices. The proposed SAMG framework allows efficient construction of nearly-linear sized graph Laplacians for coarse-level problems while maintaining good spectral approximation during the AMG setup phase by leveraging a scalable spectral graph sparsification engine. Our experimental results show that the proposed method can offer more scalable performance than existing graph-theoretic AMG solvers for solving large SDD matrices in integrated circuit (IC) simulations, 3D-IC thermal analysis, image processing, finite element analysis as well as data mining and machine learning applications.

Finally, the spectral methods are applied to power grid and thermal integrity verification applications. This dissertation introduces a vectorless power grid and thermal integrity verification framework that allows computing worst-case voltage drop or thermal profiles across the entire chip under a set of local and global workload (power density) constraints. To address the computational challenges introduced by the large 3D mesh-structured thermal grids, we apply the spectral graph reduction approach for highly-scalable vectorless thermal (or power grids) verification of large chip designs. The effectiveness and efficiency of our approach have been demonstrated through extensive experiments.

Chapter 1

Introduction

Recent research shows that by leveraging the key spectral properties of the graph Laplacians, like eigenvalues and eigenvectors, more efficient algorithms can be developed for tackling many graph-related computing tasks [97]. For example, spectral methods can potentially lead to much faster algorithms for solving sparse matrices [53, 116], numerical optimization [12], data mining [78, 103], graph analytics [39, 49], machine learning [18, 19], as well as very-large-scale integration (VLSI) computer-aided design (CAD) [27, 28, 110, 112, 113, 116]. To this end, spectral sparsification of graphs has been extensively studied in the past decade [5, 56, 93, 94, 111] to generate almost-linear-sized [1] subgraphs or sparsifiers that can robustly preserve the spectrum

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

of the original graph Laplacian. The sparsified graphs retain the same set of vertices but much fewer edges, which can be regarded as ultra-sparse graph proxies and have been leveraged for developing a series of nearly-linear-time numerical and graph algorithms [11, 32, 92, 93, 109]. Another way of simplifying graphs is to directly reduce the size of the graphs, which is widely used in many areas, including graph partitioning [43], machine learning [18] and multigrid solvers [51, 64]. However, most of the graph coarsening techniques cannot guarantee the preservation of the spectral properties on the reduced graphs, and much remains to be understood about the effect of the graph coarsening on the spectrum of a general graph.

1.1 Effective-Resistance Preserving Spectral Reduction of Graphs

In this work, we introduce a scalable algorithmic framework for *spectral reduction of graphs* for dramatically reducing the size (both nodes and edges) of undirected graphs while preserving the key spectral (structural) properties of the original graph [114, 117, 118]. The spectrally-reduced graphs will immediately lead to the development of much faster numerical and graph-related algorithms. For example, spectrally-reduced social (data) networks may allow for more efficiently modeling, mining, and analysis of large social (data) networks, spectrally-reduced neural networks allow for more

scalable model training and processing in emerging machine learning tasks, spectrally-reduced circuit networks may lead to more efficient simulation, optimization and verification of large integrated circuit (IC) systems, etc.

Our approach consists of three key phases: 1) a scalable spectrum-preserving node aggregation (reduction) phase, 2) a spectral graph sparsification phase with iterative subgraph scaling, and 3) an effective-resistance preserving post-scaling phase. To achieve truly scalable (nearly-linear time) performance for spectral graph reduction, we leverage recent similarity-aware spectral graph sparsification method [28], graph-theoretic algebraic multigrid (AMG) Laplacian solver [64, 116] and a novel constrained stochastic gradient descent (SGD) optimization approach. The major contribution of this work is summarized as follows:

(1) To well-preserve the key spectral properties of the original graph in the reduced graph, a nearly-linear time spectrum-preserving node aggregation (reduction) scheme is proposed for robustly constructing reduced graphs that have much less number of nodes.

(2) A scalable framework for spectral graph sparsification and iterative subgraph scaling is introduced for assuring sparsity of the reduced graphs by leveraging a novel constrained SGD optimization approach.

(3) We introduce a simple yet effective procedure for refining solutions, such as the Laplacian eigenvectors, computed with spectrally-reduced graphs, which immediately allows using much smaller graphs in many numerical and graph algorithms while

achieving superior solution quality.

(4) In addition, multilevel frameworks that allow us to leverage spectrally-reduced graphs for much faster spectral graph partitioning as well as t-distributed Stochastic Neighbor Embedding (t-SNE) of large data sets are proposed.

(5) We have obtained very promising experiment results for a variety of graph problems: the spectrally-reduced graphs allow us to achieve up to $1100X$ speedup for spectral graph partitioning and up to $60X$ speedup for t-SNE visualization of large data sets.

1.2 SAMG: Sparsified Graph-Theoretic Algebraic Multigrid for Solving Large Symmetric Diagonally Dominant (SDD) Matrices

Laplacian matrices of graphs arise in many numerical computational applications and graph algorithms, such as solving Symmetric Diagonally Dominant (SDD) matrices of resistive networks and elliptic partial differential equations discretized on unstructured grids [51, 89, 92, 94], spectral graph partitioning and data clustering problems [92], semi-supervised learning (SSL) [119], as well as interior-point problems for maximum flow of undirected graphs [11].

To leverage graph Laplacian matrices for developing more scalable yet reliable SDD matrix solvers, a series of graph-theoretic Algebraic Multigrid (AMG) algorithms have been proposed, such as the Combinatorial Multigrid (CMG) solver [52] and the Lean Algebraic Multigrid (LAMG) solver [64]. One common feature of these multigrid algorithms is that they construct the coarse-level problems by taking advantage of one or more properties of the graph Laplacian matrices. For instance, in the CMG solver, coarse-level nodes are formed by partitioning the graph defined by the Laplacian matrix into high conductance clusters [52], whereas the LAMG solver applies node elimination and node aggregation to form the coarse level problems [64]. Although such graph-theoretic AMG algorithms can significantly improve the efficiency and scalability for solving large SDD matrix problems over traditional direct and iterative methods, the graph based AMG operations can be potentially hindered by the dramatically increased graph densities at coarse levels. For example, one key step in the LAMG algorithm is to eliminate low-degree nodes to form a significantly smaller coarse level problem (usually 4X node reduction over the finer level), which usually leads to a dramatically increased number of elements at the coarse level if there are too many high-degree nodes; a similar step in CMG is to cluster the graph into well-connected parts to form the coarse-level nodes, which may not be possible if the graph itself is already very dense, such as the k-nearest neighbor graphs (k-NNGs) that have been heavily studied in data mining and machine learning communities.

To address the challenges in existing graph-theoretic AMG algorithms for solving large

SDD matrices, we propose a Sparsified graph-theoretic Algebraic Multigrid (SAMG) algorithm [116], by introducing a spectral graph sparsification procedure during the SAMG setup phase for creating coarse level problems. We show that by leveraging a recent spectral-perturbation based graph sparsification method [27], ultra sparse coarse level problems (graphs) can be reliably constructed without loss of spectral similarity with the original coarse problems (graphs). In other words, coarse level problems created with the proposed SAMG framework are always ultra sparse yet spectrally similar to the original problem, leading to highly efficient yet robust AMG algorithms for solving large SDD matrices. Our results show that the proposed SAMG framework is able to further improve the scalability of existing graph-theoretic AMG methods that are already very efficient.

1.3 A Spectral Approach to Scalable Vectorless Power Grid and Thermal Integrity Verification

Aggressive VLSI technology scaling has led to dramatically increased power densities as well as significantly elevated temperature on-chip, which imposes ever-increasing challenges in designing integrated circuit (IC) systems [76]. For example, the power distribution network of an IC design must be verified throughout the design process

to ensure the supply voltage fluctuations are within certain thresholds. However, the increased chip power dissipation and reduced supply voltages result in a massive amount of the current drawn from the power supply, which is further distributed over the much larger power grids, making power grid verification increasingly challenging and critical for robust chip designs. The increased temperatures will result in: (1) larger power grid IR drops and interconnect RC delays due to the increased interconnect resistivity; (2) higher leakage power influenced by the exponential increasing of sub-threshold current; (3) slower devices because of the degraded carrier mobility; (4) shorter device life and poor package reliability by the potentially existence of local hot spots as well as unevenly temperature distribution across the die. To achieve the desired level of chip reliability and functionality, compute-intensive full-chip thermal analysis and verification are indispensable, which typically involves estimating thermal profiles under various kinds of workloads and power budget: at the circuit level, estimating temperature variations and peak temperature across the chip are essential for accurate timing and power analysis of digital designs [76], whereas evaluating peak temperature and temperature gradients for critical circuit modules become increasingly important for reducing mismatches and improving the performance of analog and mixed-signal circuits [62]; at a system level, thermal modeling and analysis can be leveraged to guide dynamic voltage and frequency scaling (DVFS) for reducing thermal violations, achieving desired temperature levels and thereby reducing workload runtimes [13].

Traditional vector-based power grid and thermal integrity verification rely on running numerous circuit simulations using over-pessimistic power distributions to locate the vector, which results in the worst-case voltage drop or thermal profile. In this case, It requires the underlying workloads or power densities to be known in advance [38, 60, 88], which may not always be practical. For example, at the early chip design phase, it is usually impossible to obtain accurate estimation of underlying power densities since accurate modeling for workloads may not be necessarily available at the early design phase. As a result, traditional power grid and thermal analysis methods may not always provide useful guidance for verifying and improving the design reliability and performance that can be significantly impacted by extreme (worst-case) chip power and thermal profiles, such as worst-case temperature or thermal gradients across the chip.

Due to the above limitation, vectorless integrity verification methods have been considered as alternatives, which have already been widely studied for power grid verifications. It uses the optimization approaches to find the worst-case scenarios under specific workloads constraints. A series of vectorless power grid verification techniques has been investigated in the past decade [15, 25, 26, 33, 34, 50, 70, 81, 105]. Recent research has made significant progress in reducing the power grid verification costs by using novel sparse approximate inverse (SPAI) technique [33], efficient dual algorithm [105], and node elimination [34]. Despite these significant improvements, the overall power grid verification cost is still extremely high, especially for large-scale

verification tasks.

To significantly improve the verification efficiency, scalable multilevel vectorless verification methods based on geometric multigrid (GMG) operations and PDE-constrained optimization framework have been proposed [25, 26, 58] to tackle large scale flip-chip power grid integrity verification problems. However, such methods usually require the underlying power grid designs to have relatively regular structures so that GMG operations can be performed effectively, which can become a major limitation when applied to nanoscale PDN designs where regular power grid structures are rare to find.

Motivated by the existing GMG-based multilevel vectorless verification methods, [113] introduces a more general multilevel power grid verification framework that leverages the recent graph-theoretic algebraic multigrid (AMG) algorithmic framework [64] as well as a hierarchy of almost linear-sized power grid sparsifiers, making it scalable to very large scale power grids without considering the geometric information.

Although vectorless thermal integrity verification tasks are similar to existing power grid integrity verification problems, the computational challenges can be dramatically different: in thermal verification problems, 3D mesh-structured thermal grids are involved, whereas for power grid verification tasks 2D meshes are usually considered. Existing vectorless verification methods need to set up linear programs (LPs) for finding the worst-case vectors that will lead to the extreme thermal profiles, which

requires computing thermal sensitivities with respect to each power source. However, the 3D meshes in thermal grid verification tasks are much more challenging to tackle than the 2D meshes in power grid verification [105, 107] due to the super-linear complexities of existing vectorless verification methods. Hence, the majority of the thermal analysis approaches rely on the vector-based simulations [38, 60]. Recently, [115] proposed the first vectorless thermal verification algorithm, which can be easily scaled to very large scale thermal grid designs.

Motivated by the existing vectorless integrity verification problems [25, 33, 105, 107, 113, 115], we propose the first general vectorless integrity verification framework which can be applied to both power and thermal grids to provide the scalable solutions for estimating the maximum voltage drop or the nearly-worst-case thermal profiles under various complex power density or workload uncertainties and constraints. It leverages a recent graph-theoretic algebraic multigrid (AMG) algorithmic framework [64] as well as a hierarchy of almost linear-sized sparsifiers. The proposed vectorless verification approach gains insights from prior multilevel PDE-constrained optimization methods [58], circuit adjoint sensitivity analysis, spectral graph theory [94] and emerging graph signal processing research [87]. The original multilevel optimization method assumes that once given a hierarchy of model problems ordered from the finest to the coarsest levels. The optimization solution can be incrementally improved on coarser to coarsest level problems, while the coarser level optimization solution can effectively facilitate finding the optimal solution for the original problem.

To address the computational challenges in vectorless thermal integrity verification, we propose to aggressively simplify the 3D thermal grids during vectorless verification while assuring the approximation accuracy via spectral graph sparsification and iterative edge weight scaling. To this end, motivated by recent graph signal processing research [87], we propose a mathematically rigorous method to match full-chip temperature distributions that can be understood as the "low-frequency" graph signals on thermal grids obtained after applying a "low-pass" graph filter on the original input power sources. Our thermal grid simplification task will aim to minimize the number of edges in the sparsified thermal grid that can still precisely preserve slowly-varying, "low-frequency" temperature distribution across the entire thermal grid. Such simplified thermal grids will allow finding worst-case thermal profiles in a highly efficient way without losing accuracy. The proposed vectorless thermal integrity verification method is highly scalable and thus can be adopted in either the early chip design phase or final chip verification phase. The main contribution of this work is briefly summarized as follows:

(1) We propose a general framework for vectorless power grid and thermal integrity verification that allows estimating nearly-worst-case scenarios under various kinds of complex workload or power density uncertainties and constraints.

(2) To make the proposed method scalable to large problems, we introduce a multilevel vectorless verification framework that is significantly accelerated by a novel power and thermal grid simplification method motivated by emerging spectral graph

sparsification and graph signal processing research.

(3) We demonstrate extensive experimental results on both power and thermal grids designs with various problem sizes and power density (workload) constraints, as well as the flexible tradeoffs between the verification cost and solution quality enabled by the proposed vectorless verification method.

1.4 Overview of Chapters

This dissertation includes five chapters. Chapter 1 describes three major problems and our contributions. Chapter 2 proposed a scalable spectral graph reduction framework which can well preserve the spectral properties (i.e. effective resistances) on the reduced graphs. The framework can be well applied to various applications, like graph partitioning and data visualizations. Chapter 3 introduces a sparsified algebraic multigrid solver for solving large symmetric diagonally dominant Laplacian matrices. Chapter 4 introduces the framework for vectorless power grid and thermal integrity verification that allows estimating nearly-worst-case scenarios under various kinds of complex workloads or power density uncertainties and constraints. Chapter 5 presents the conclusions of this dissertation and discusses the possible future works to be explored.

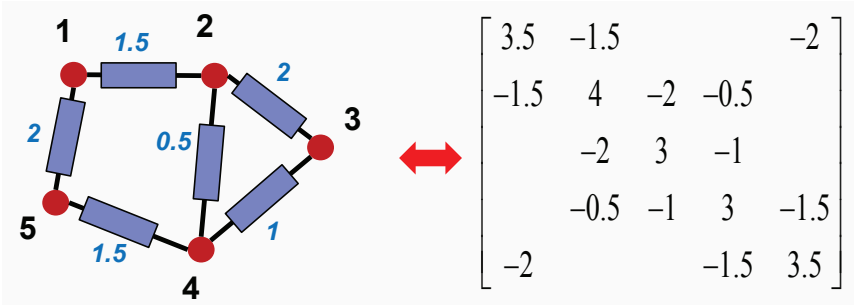


Figure 1.1: A resistor network (conductance value of each element is shown) and its graph Laplacian matrix.

1.5 Preliminaries

1.5.1 Laplacian Matrices of graphs

Consider an undirected graph $G = (V, E_G, w_G)$ with V denoting the set of vertices, E_G denoting the set of undirected edges, and w_G denoting the associated edge weights. We define $\mathbf{D_G}$ to be a diagonal matrix with $D_G(i,i)$ being equal to the (weighted) degree of node i, and $\mathbf{A_G}$ and $\mathbf{L_G}$ to be the adjacency and Laplacian matrices of undirected graph G as follows, respectively:

$$\mathbf{A_G}(i,j) = \begin{cases} w_G(i,j) & \text{if } (i,j) \in E_G \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

Graph Laplacians can be constructed by using $\mathbf{L_G} = \mathbf{D_G} - \mathbf{A_G}$ and will satisfy the following conditions: **1.** Each column and row sum will be equal to zero; **2.** All off-diagonal elements are non-positive; **3.** The graph Laplacian is a symmetric diagonally dominant (SDD) matrix, which can be considered as an admittance matrix of a resistive circuit network [89], as is shown in Figure 1.1

1.5.2 Spectral Sparsification of Graphs

To further push the limit of spectral methods for handling big (data) graphs, spectral graph theory has been extensively studied by mathematics and theoretical computer science (TCS) researchers in the past decade [5, 14, 47, 48, 56, 78, 93, 94]. 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 first few eigenvalues and eigenvectors of the graph Laplacian. The related results have lead to the development of a variety of *nearly-linear time* numerical and graph algorithms for solving large sparse matrices, graph-based semi-supervised learning (SSL), computing the stationary distributions of Markov chains and personalized PageRank vectors, spectral graph partitioning, data clustering, max-flow of undirected graphs, etc [11, 14, 32, 48, 51, 89, 91, 92, 93, 94].

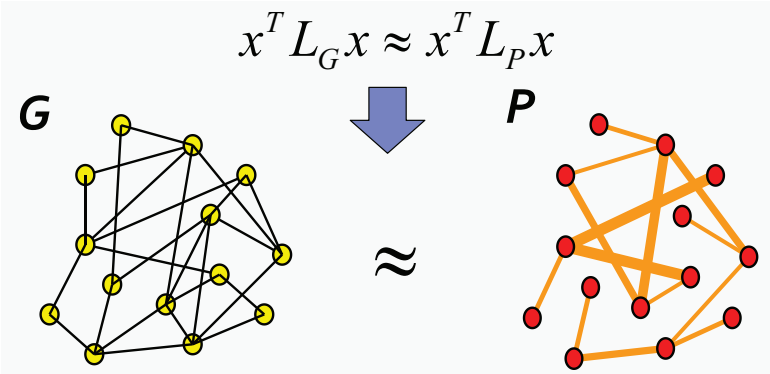


Figure 1.2: Two spectrally similar graphs G and P.

Spectral graph sparsification aims to find a spectrally-similar subgraph (sparsifier) $P = (V, E_P, w_P)$ that has the same set of vertices of the original graph $G = (V, E_G, w_G)$, but much fewer edges, as shown in Figure 1.2. There are two types of sparsification methods: the cut sparsification methods preserve the cut value through random sampling of edges [6], whereas spectral sparsification methods preserve the graph spectral (structural) properties, such as distances between vertices, cuts in the graph, as well as the stationary distributions of Markov chains [14, 94]. Therefore, spectral graph sparsification is a much stronger notion than cut sparsification. We say G and its subgraph P are $\sigma-$spectrally similar if the following condition holds for all real vectors $\mathbf{x} \in \mathbb{R}^V$:

$$\frac{\mathbf{x}^\top \mathbf{L}_P \mathbf{x}}{\sigma} \leq \mathbf{x}^\top \mathbf{L}_G \mathbf{x} \leq \sigma \mathbf{x}^\top \mathbf{L}_P \mathbf{x}, \tag{1.2}$$

where $\mathbf{L}_G$ and $\mathbf{L}_P$ denote the Laplacian matrices of graph G and P, respectively. Define the relative condition number as $\kappa(\mathbf{L}_G, \mathbf{L}_P) = \lambda_{\max}/\lambda_{\min}$, where $\lambda_{\max}$ and $\lambda_{\min}$

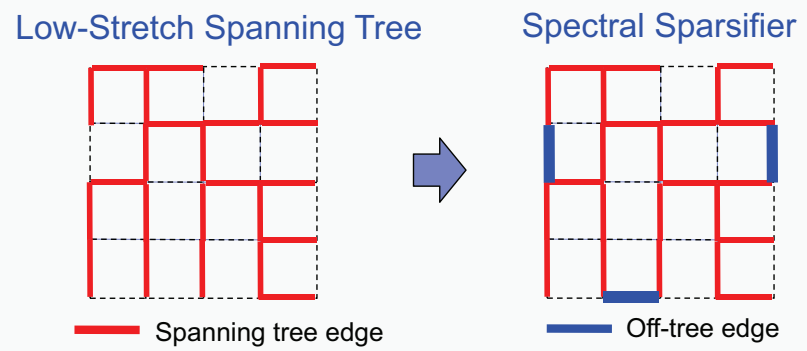


Figure 1.3: A spanning tree and its ultra-sparsifier subgraph.

are the largest and smallest nonzero eigenvalues of

$$\mathbf{L}_G\mathbf{u} = \lambda\mathbf{L}_P\mathbf{u}, \tag{1.3}$$

where $\mathbf{u}$ is the generalized eigenvector of $\mathbf{L_G}$. It can be further shown that $\kappa(\mathbf{L}_G, \mathbf{L}_P) \leq \sigma^2$, which indicates that a smaller relative condition number or σ^2 corresponds to a higher spectral similarity.

For complete graphs, Ramanujan graphs are the best spectral sparsifiers, whereas for general graphs the Twice-Ramanujan sparsifiers are the best but will need $O(mn^3)$ time for constructing sparsifiers, where $m = |E|$ and $n = |V|$ [5]. For general graphs, the state-of-the-art nearly-linear time spectral sparsification methods rely on extracting Low-Stretch Spanning Trees (LSSTs) that have been key to the development of nearly-linear time algorithms for solving SDD matrices [27, 48, 51, 89, 90, 92], which typically involve the following steps: 1) extracting a low-stretch spanning tree from the original graph to form the backbone of the graph sparsifier; 2) recovering

"spectrally critical" off-tree edge to the spanning tree to form an ultra-sparse graph sparsifier, as shown in Figure 1.3. To this end, an effective-resistance based edge sampling scheme and spectral perturbation based edge selection scheme have been proposed for recovering these off-tree edges [27, 90], which leads to the development of much faster SDD matrix solvers [116] and spectral graph partitioning algorithm [28]. Although both methods have a worst-case nearly-linear time complexity, the spectral perturbation based approach is considered more practically efficient for dealing with general large networks.

Chapter 2

Effective-Resistance Preserving Spectral Reduction of Graphs

2.1 Background

There are two major ways to simplify a graph: graph sparsification aims to reduce the number of edges, while graph coarsening reduces the number of graph nodes. Graph sparsification and coarsening have been widely used in the applications of graph clustering and partitioning [20, 43, 84, 102], as well as data (graph) visualization [35, 45, 101]

Different graph sparsification techniques have been proposed. Graph spanners [24, 77]

were proposed to preserve the pair distances between nodes. Bencztir and Kargert [6, 7] then introduced the cut sparsifier, which can preserve cut values between the original graph and the sparsified graph. Later, Spielman and Teng [94] proposed the spectral sparsifier for preserving the key eigenvalues and eigenvectors, which is a stronger notation than the cut sparsifier. Since then, more spectral related sparsification methods are proposed, like the spectral preservation of pseudoinverse for the graph Laplacian by Li [59].

Compared to the solid theoretical work on the graph sparsification, graph coarsening is harder to understand due to the lack of mature theoretical frameworks. A variety of spectral coarsening schemes have been proposed, but the majority of the algorithms are based on heuristics. [23] proposed the Kron reduction of the graph based on the Schur complement. Purohit et al. [80] introduced the CoarseNet that is able to coarsen while preserving the largest eigenvalue of its adjacency matrix, such that the diffusion characteristics of the original graph can be kept. Loukas and Vandergheynst [66, 67] proposed a theoretical framework that proves the spectral preservation of the original graph after coarsening based on the concept of the restricted spectral similarity. Recently, Bravo-Hermsdorff and Gunderson [36] proposed a unified framework of graph sparsification and coarsening which aims to preserve the Laplacian pseudoinverse on the reduced graph.

2.2 Spectral Reduction of Graphs

2.2.1 Overview

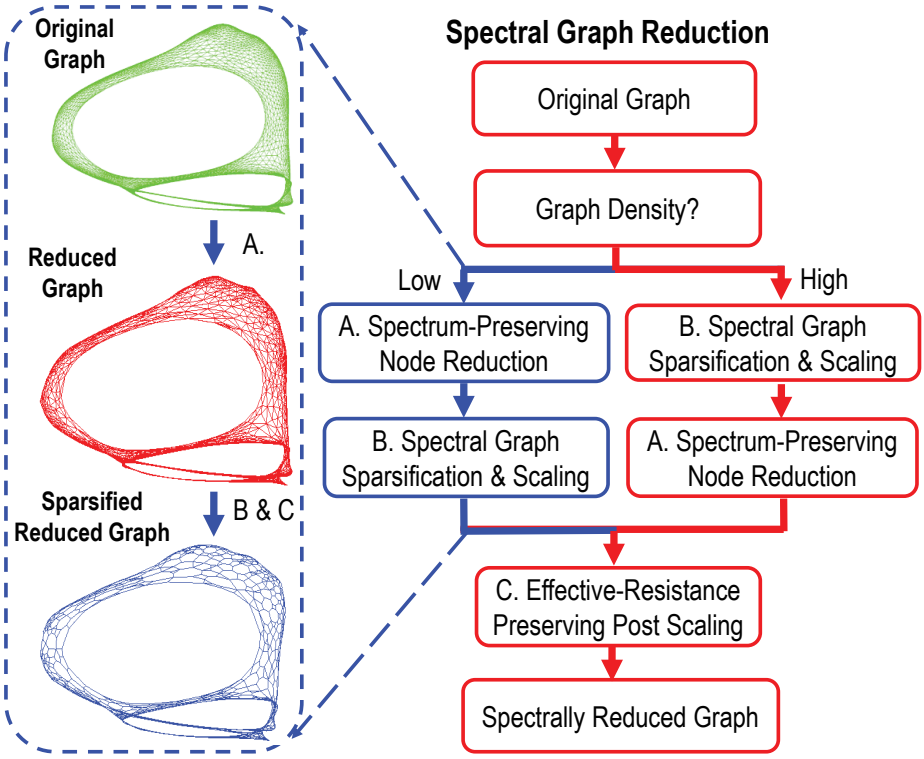


Figure 2.1: The proposed spectral reduction framework.

In the following, assume that $G = (V, E_G, w_G)$ is a weighted, undirected, and connected graph, $P = (V, E_P, w_P)$ is the spectrally sparsified graph of G, $R = (V_R, E_R, w_R)$ is the reduced graph of G without sparsification, and $S = (V_R, E_S, w_S)$ is the sparsified reduced graph of G. The Laplacian matrices of the corresponding graphs have been shown in Table 2.1 that also includes the fine-to-coarse (G-to-R)

Table 2.1
Symbols and their denotations in this work

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

graph mapping matrix denoted by $\mathbf{H_G^R}$ as well as the coarse-to-fine (R-to-G) graph mapping matrix denoted by $\mathbf{H_R^G}$.

This work introduces a *spectral graph reduction* framework (as shown in Figure 2.1) that allows computing much smaller yet spectrally-similar graph S such that the following condition holds for all real vectors $\mathbf{x} \in \mathbb{R}^{\mathbb{V}}$:

$$\frac{\mathbf{x_R}^\top \mathbf{L}_S \mathbf{x_R}}{\sigma} \leq \mathbf{x}^\top \mathbf{L}_G \mathbf{x} \leq \sigma \mathbf{x_R}^\top \mathbf{L}_S \mathbf{x_R}, \quad \mathbf{x_R} = \mathbf{H_G^R x}. \tag{2.1}$$

An overview of the proposed method for spectral reduction of large graphs is described as follows. Our approach for spectral reduction of undirected graphs includes the following two phases: **Phase (A)** will determine the fine-to-coarse graph mapping operator using spectral node proximity measurement computed based on algebraic distance [10], and reduce the original graph into a much smaller graph using the fine-to-coarse graph mapping operator; **Phase (B)** will extract spectrally-similar sparsifiers of the original (reduced) graph and scale up edge weights in the sparsified

graphs to better match the key spectral (structural) properties, such as the eigenvalues/eigenvectors of graph Laplacians. **Phase (C)** will globally scale up the sparsified (reduced) graph for matching the original Laplacian eigenvalues and effective resistances between nodes. Since the spectral node proximity metric based on algebraic distance cannot be directly applied to dense graphs [10], our approach will first examine the average node degrees in the original graph: if the original graph is relatively sparse ($|E_G| < 40|V|$), **Phases (A) to (C)** will be performed in sequence as shown in Figure 2.1; otherwise, if the original graph is too dense ($|E_G| > 40|V|$), **Phase (B)** for spectral graph sparsification and edge scaling will be performed first, which is followed by **Phase (A)** and **Phase (C)**.

2.2.2 Phase (A): Spectrum-Preserving Node Reduction

2.2.2.1 Spectral node affinity metric.

To generate the reduced graph based on the original graph, a spectrum-preserving node aggregation scheme is applied based on spectral node affinity $a_{p,q}$ defined as follows for neighboring nodes p and q [10, 64]:

$$a_{p,q} = \frac{\|(\mathbf{X_{p,:}}, \mathbf{X_{q,:}})\|^{\mathbf{2}}}{(\mathbf{X_{p,:}}, \mathbf{X_{p,:}})(\mathbf{X_{q,:}}, \mathbf{X_{q,:}})}, (\mathbf{X_{p,:}}, \mathbf{X_{q,:}}) = \mathbf{\Sigma_{k=1}^{K}}(\mathbf{x_p^{(k)}} \cdot \mathbf{x_q^{(k)}}) \tag{2.2}$$

where $\mathbf{X} = (\mathbf{x^{(1)}}, \ldots, \mathbf{x^{(K)}})$ includes K test vectors computed by applying a few Gauss-Seidel (GS) relaxations for solving the linear system of equations $\mathbf{L_G x^{(i)}} = 0$ for $i = 1, ..., K$ with K random vectors that are orthogonal to the all-one vector $\mathbf{1}$ or equivalently satisfying $\mathbf{1}^\top \mathbf{x^{(i)}} = \mathbf{0}$. Let $\tilde{\mathbf{x}}^{(\mathbf{i})}$ denote the approximation of the true solution $\mathbf{x^{(i)}}$ after applying several GS relaxations to $\mathbf{L_G x^{(i)}} = 0$. Due to the smoothing property of GS relaxation, the latest error can be expressed as $\mathbf{e_s^{(i)}} = \mathbf{x^{(i)}} - \tilde{\mathbf{x}}^{(\mathbf{i})}$, which will only contain the smooth (low-frequency) modes of the initial error, while the oscillatory (high-frequency) modes of the initial error will be effectively eliminated [8]. Based on the K smoothed vectors in $\mathbf{X}$, it is possible to embed each node into a K-dimensional space such that node p and node q are considered spectrally-close enough to each other if their low-dimensional embedding vectors $\mathbf{x_p} \in \mathbb{R}^K$ and $\mathbf{x_q} \in \mathbb{R}^K$ are highly correlated. Spectrally-similar nodes p and q can be then aggregated together for node reduction purpose.

2.2.2.2 Spectral similarity between nodes.

It has been shown that the node affinity metric $a_{p,q}$ can usually effectively reflect the distance or strength of connection between nodes p and q in a graph [64]: a larger $a_{p,q}$ value indicates a stronger spectral similarity (correlation) between nodes p and q. Consequently, the nodes with large affinity should be aggregated together to form the nodes in the reduced graph. Once node aggregation schemes are determined, the

graph mapping operators ($\mathbf{H}_{\mathbf{G}}^{\mathbf{R}}$ and $\mathbf{H}_{\mathbf{R}}^{\mathbf{G}}$) can be obtained and leveraged for constructing spectrally-reduced graphs. For example, the reduced Laplacian can be computed by $\mathbf{L_R} = \mathbf{H}_{\mathbf{G}}^{\mathbf{R}}\mathbf{L_G}\mathbf{H}_{\mathbf{R}}^{\mathbf{G}}$, which uniquely defines the reduced graph.

We emphasize that the node aggregation (reduction) scheme based on the above spectral node affinity calculations will have a (linear) complexity of $O(|E_G|)$ and thus allow preserving the spectral (global or structural) properties of the original graph in the reduced graph in a highly efficient and effective way: the smooth components in the first few Laplacian eigenvectors can be well-preserved after node aggregation, which is key to preserving the first few (bottom) eigenvalues and eigenvectors of the original graph Laplacian in the reduced graphs [68].

2.2.2.3 Limitations when dealing with dense graphs.

The above node reduction scheme based on the algebraic distance metric may not be reliable when applied to dense graph problems. Since each node in the dense graph will typically connect to many other nodes, running a few GS relaxations will result in many nodes seemingly close to each other and can lead to rather poor node aggregation results. For example, an extreme case is to directly apply the above node aggregation scheme to a complete graph where each node has $|V|-1$ edges connecting to the rest of the nodes: since applying GS relaxations will immediately assign the

same values to all nodes, no meaningful clusters of nodes can be identified. As shown in our experiment results, it is not possible to use the above node affinity metric for aggregating nodes for the "appu" graph [17] that has high average node degrees ($|E_G|/|V| \approx 90$).

To this end, we propose to perform a spectral sparsification and scaling procedure (**Phase (B)**) before applying the node aggregation (reduction) phase. Such a scheme will allow extracting ultra-sparse yet spectrally-similar subgraphs and subsequently aggregate nodes into clusters using the above node affinity metric. As a result, the spectral graph reduction flow proposed in this work can be reliably applied to handle both sparse and dense graphs, as shown in Figure 2.1.

2.2.3 Phase (B): Spectral Graph Sparsification and Scaling

2.2.3.1 Copping with high graph densities.

The proposed spectral node aggregation scheme in Section 2.2.2 will enable us to reliably construct smaller graphs that have fewer vertices. However, the aggregated nodes may potentially result in much denser graphs (with significantly higher node degrees), which may incur even greater computational and memory cost for graph operations. For example, emerging multi-way spectral graph partitioning (clustering)

algorithms [55, 78] are required to compute multiple Laplacian eigenvectors, which can still be very costly for dense graphs since the efficiency of modern eigensolvers or eigendecomposition methods strongly depend on the matrix sparsity [57, 83, 106].

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

2.2.3.2 Spectral approximation with spanning tree subgraphs.

Denote the total stretch of the spanning-tree subgraph P with respect to the original graph G to be $\mathrm{st}_P(G)$, which can be calculated by the following formular:

$$\mathrm{st}_P(G) = \sum_{e \in G \setminus P} |c_e| \tag{2.3}$$

where c_e is the unique cycle in $T \cup \{e\}$. Spielman showed that $\mathbf{L}_P^{+}\mathbf{L}_G$ has at most k generalized eigenvalues greater than $\mathrm{st}_P(G)/k$ [95]. It has been shown that every graph has a low-stretch spanning tree (LSST) with bounded total stretch [95], which

leads to:

$$\kappa(\mathbf{L}_G, \mathbf{L}_P) \leq \mathrm{Tr}(\mathbf{L}_P^{+}\mathbf{L}_G) = \mathrm{st}_P(G) \leq (m \log n \log\log n), \tag{2.4}$$

where $m = |E_G|$, $n = |V|$, and $\mathrm{Tr}(\mathbf{L}_P^{+}\mathbf{L}_G)$ is the trace of $\mathbf{L}_P^{+}\mathbf{L}_G$. Such a result motivates to construct an ultra-sparse yet spectrally-similar subgraphs by recovering only a small portion of important off-tree edges to the spanning tree. For example, a recent spectral perturbation framework [27, 28] allows constructing the σ-similar spectral sparsifiers with $O(m \log n \log\log n/\sigma^2)$ off-tree edges in nearly-linear time.

2.2.3.3 Towards better approximation with off-tree edges.

To reduce the spectral distortion between the original graph and the spanning tree, a spectral off-tree edge embedding scheme and edge filtering method with approximate generalized eigenvectors have been proposed in [27, 28], which is based on following spectral perturbation analysis:

$$\mathbf{L}_G(\mathbf{u}_i + \delta\mathbf{u}_i) = (\lambda_i + \delta\lambda_i)(\mathbf{L}_P + \delta\mathbf{L}_P)(\mathbf{u}_i + \delta\mathbf{u}_i), \tag{2.5}$$

where a perturbation $\delta\mathbf{L}_P$ is applied to $\mathbf{L}_P$, 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 [27] leads to:

$$-\frac{\delta\lambda_i}{\lambda_i} = \mathbf{u}_i^\top \delta\mathbf{L}_P \mathbf{u}_i, \tag{2.6}$$

which indicates that the reduction of λ_i is proportional to the Laplacian quadratic form of $\delta\mathbf{L}_P$ with the generalized eigenvector $\mathbf{u}_i$. Therefore, if the dominant eigenvector $\mathbf{u}_n$ is applied, the largest generalized eigenvalue λ_n can be significantly reduced by properly choosing $\delta\mathbf{L}_P$ that includes the set of off-tree edges and their weights. Once the largest generalized eigenvalue becomes sufficiently small, the distortion between subgraph P and the original graph G will be greatly reduced.

An alternative view of such a spectral embedding scheme is to consider the following **Courant-Fischer theorem** for generalized eigenvalue problems:

$$\lambda_n = \max_{\substack{|\mathbf{x}|\neq 0 \\ \mathbf{x}^\top\mathbf{1}=0}} \frac{\mathbf{x}^\top\mathbf{L}_G\mathbf{x}}{\mathbf{x}^\top\mathbf{L}_P\mathbf{x}} \approx \max_{\substack{|\mathbf{x}|\neq 0 \\ x(p)\in\{0,1\}}} \frac{\mathbf{x}^\top\mathbf{L}_G\mathbf{x}}{\mathbf{x}^\top\mathbf{L}_P\mathbf{x}} = \max \frac{|\partial_G(Q)|}{|\partial_P(Q)|}, \tag{2.7}$$

where $\mathbf{1}$ is the all-one vector, the node set Q is defined as

$$Q \overset{\text{def}}{=} \{p \in V : x(p) = 1\}, \tag{2.8}$$

and the boundary of Q in G is defined as

$$\partial_G(Q) \overset{\text{def}}{=} \{(p,q) \in E_G : p \in Q, q \notin Q\}, \tag{2.9}$$

which will lead to

$$\begin{aligned} \mathbf{x}^\top \mathbf{L}_G \mathbf{x} &= |\partial_G(Q)|, \\ \mathbf{x}^\top \mathbf{L}_P \mathbf{x} &= |\partial_P(Q)|. \end{aligned} \tag{2.10}$$

According to (2.7), $\lambda_{max} = \lambda_n$ will reflect the largest mismatch of the boundary (cut) size between G and P, since finding the dominant generalized eigenvector is similar to finding the node set Q such that $\frac{|\partial_G(Q)|}{|\partial_P(Q)|}$ or the mismatch of boundary (cut) size between the original graph G and subgraph P is maximized. Once Q or $\partial_P(Q)$ can be identified by spectral graph embedding using dominant generalized eigenvectors, the edges in $\partial_G(Q)$ can be selected and recovered to P to reduce the maximum mismatch or λ_n.

Denote $\mathbf{e}_p \in \mathbb{R}^V$ to be the elementary unit vector with only the p-th element being 1 and others being 0, and we denote $\mathbf{e}_{p,q} = \mathbf{e}_p - \mathbf{e}_q$. Then by including the off-tree edges, the generalized eigenvalue perturbation can be expressed as follows:

$$-\frac{\delta\lambda_i}{\lambda_i} = \mathbf{u}_i^\top \delta\mathbf{L}_{P,max}\mathbf{u}_i = \sum_{(p,q)\in E_G \setminus E_P} w_G(p,q)\left(\mathbf{e}_{p,q}^T \mathbf{u}_i\right)^2, \tag{2.11}$$

where $\delta\mathbf{L}_{P,max} = \mathbf{L}_G - \mathbf{L}_P$, and $w_G(p,q)$ denotes the weight of edge (p,q) in the original graph. The **spectral criticality** $c_{p,q}$ of each off-tree edge (p,q) is defined as:

$$c_{p,q} = w_G(p,q)\left(\mathbf{e}_{p,q}^T \mathbf{u}_n\right)^2. \tag{2.12}$$

If we consider the undirected graph G to be a resistor network, and $\mathbf{u}_n$ to be the voltage vector for that resistor network, $c_{p,q}$ can be regarded as the edge Joule heat (power dissipation). Consequently, the most spectrally critical off-tree edges from $\partial_G(Q)$ can be identified and recovered into LSST for spectral graph topology sparsification by (2.12), which allows improving spectral approximation in the subgraph by dramatically reducing the λ_n. In practice, approximate generalized eigenvectors computed through a small number of generalized power iterations will suffice for low-dimensional spectral off-tree edge embedding, which can be realized as follows:

(1) Compute an approximate eigenvector $\mathbf{h}_t$ by applying t-step generalized power iterations on an initial vector $\mathbf{h}_0 = \sum_{i=1}^{n} \alpha_i \mathbf{u}_i$:

$$\mathbf{h}_t = \left(\mathbf{L}_P^{+}\mathbf{L}_G\right)^t \mathbf{h}_0 = \left(\sum_{i=1}^{n} \lambda_i \mathbf{u}_i \mathbf{u}_i^T\right)^t \sum_{i=1}^{n} \alpha_i \mathbf{u}_i = \sum_{i=1}^{n} \alpha_i \lambda_i^t \mathbf{u}_i; \tag{2.13}$$

(2) Compute the quadratic form for off-tree edges with $\mathbf{h_t}$:

$$\begin{aligned} -\tfrac{\delta\lambda_i}{\lambda_i} &\approx \mathbf{h_t}^{\top} \delta \mathbf{L}_{P,max} \mathbf{h_t} = \sum_{i=1}^{n} (\alpha_i \lambda_i^t)^2 (\lambda_i - 1) \\ &= \sum_{(p,q)\in E_G \setminus E_P} w_G(p,q) \left(\mathbf{e}_{p,q}^T \mathbf{h}_t\right)^2 = \sum_{(p,q)\in E_G \setminus E_P} \tilde{c}_{p,q}, \end{aligned} \tag{2.14}$$

where $\tilde{c}_{p,q}$ denotes the approximate spectral criticality of each off-tree edge (p,q). It should be noted that using k vectors computed by (2.13) will enable to embed

each node into a k-dimensional **generalized eigenspace**, which can facilitate edge filtering from $\partial_G(Q)$ to avoid recovering similar edges into P. In this work, we choose $t = 2$, which already leads to consistently good results for a large variety of graph problems. To achieve more effective edge filtering for similarity-aware spectral graph sparsification, an incremental graph densification procedure [28] will be adopted in this work. During each graph densification iteration, a small portion of "filtered" off-tree edges will be added to the latest spectral sparsifier, while the spectral similarity is estimated to determine if more off-tree edges are needed.

2.2.3.4 Subgraph scaling via constrained optimization.

To aggressively limit the number of edges in the subgraph P while still achieving a high quality approximation of the original graph G, we propose an efficient spectral scaling scheme for scaling up edge weights in the subgraph P to further reduce the largest mismatch or λ_n. The dominant eigenvalue perturbation $\delta\lambda_n$ can be expressed in terms of edge weight perturbations as follows:

$$-\frac{\delta\lambda_n}{\lambda_n} = \mathbf{u}_n^\top \delta\mathbf{L}_P \mathbf{u}_n = \sum_{(p,q)\in E_P} \delta w_P(p,q) \left(\mathbf{e}_{p,q}^\top \mathbf{u}_n\right)^2, \tag{2.15}$$

which directly gives the sensitivity of λ_n with respect to each edge weight $w_P(p,q)$ in graph P:

$$\frac{\delta\lambda_n}{\delta w_P(p,q)} = -\lambda_n \left(\mathbf{e}_{p,q}^\top \mathbf{u}_n\right)^2 \approx -\lambda_n \left(\mathbf{e}_{p,q}^\top \mathbf{h}_t\right)^2. \tag{2.16}$$

The (approximate) sensitivity expressed in (2.16) can be leveraged for finding a proper edge weight scaling factor for each edge in P such that λ_n will be reduced. Since scaling up edge weights in P will result in the monotonic decrease of both λ_n and λ_1, it is likely that λ_1 will decrease at a faster rate than λ_n, which leads to a degraded spectral similarity between G and P. To avoid such a degradation in spectral approximation quality, we propose the following methods for estimating the extreme generalized eigenvalues λ_n and λ_1, which allows us to more properly scale-up edge weights in P.

The largest eigenvalues of $\mathbf{L}_P^+\mathbf{L}_G$ are well separated from each other [95]; hence, we can accurately calculate the largest eigenvalue (λ_n) by performing only a small number of generalized power iterations with an initial random vector. Since the generalized power iterations can converge at a geometric rate governed by λ_{n-1}/λ_n, the error of the estimated largest generalized eigenvalue will drop to $|\lambda_{n-1}/\lambda_n|^k e_0$ after k iterations for an initial error e_0. As a result, only a few (five to ten) iterations will be sufficient to compute a good estimation of λ_n for well-separated largest eigenvalues that lead to small λ_{n-1}/λ_n. To gain scalable runtime performance, we will leverage recent graph-theoretic algebraic multigrid (AMG) algorithms for solving the sparsified Laplacian

matrix $\mathbf{L_P}$ [64, 116].

Since the smallest eigenvalues of $\mathbf{L}_P^+\mathbf{L}_G$ are crowded together, using (shifted) inverse power iterations may not be efficient due to the slow convergence caused by relatively poor separation of smallest eigenvalues. To more efficiently estimate the smallest generalized eigenvalue, we leverage the Courant-Fischer theorem for approximately computing the smallest generalized eigenvalues:

$$\lambda_1 = \lambda_{min} = \min_{\substack{|\mathbf{x}|\neq 0 \\ \mathbf{x}^\top \mathbf{1}=0}} \frac{\mathbf{x}^\top \mathbf{L}_G \mathbf{x}}{\mathbf{x}^\top \mathbf{L}_P \mathbf{x}}, \tag{2.17}$$

which indicates that the key to locating the smallest generalized eigenvalues is to find a vector $\mathbf{x}$ that minimizes the ratio between the quadratic forms of the original and sparsified Laplacians. In our method, we will require every element in $\mathbf{x}$ to only take a value 1 or 0 for each node in both G and P for minimizing the following ratio, which will lead to a good estimation for λ_1:

$$\lambda_1 \approx \min_{\substack{|\mathbf{x}|\neq 0 \\ x(p)\in\{0,1\}}} \frac{\mathbf{x}^\top \mathbf{L}_G \mathbf{x}}{\mathbf{x}^\top \mathbf{L}_P \mathbf{x}} = \min_{\substack{|\mathbf{x}|\neq 0 \\ x(p)\in\{0,1\}}} \frac{\sum\limits_{x(p)\neq x(q),(p,q)\in E_G} w_G(p,q)}{\sum\limits_{x(p)\neq x(q),(p,q)\in E_P} w_P(p,q)}, \tag{2.18}$$

To this end, we initialize all nodes with the same value of 0 and only select a single node p to be assigned with a value of 1, which leads to:

$$\lambda_1 \approx \min_{p\in V} \frac{\mathbf{d}_G(p)}{\mathbf{d}_P(p)}, \tag{2.19}$$

where $\mathbf{d}_G$ and $\mathbf{d}_P$ are the diagonal vectors of $\mathbf{L_G}$ and $\mathbf{L_P}$ satisfying $\mathbf{d}_G(p) = \mathbf{L}_G(p,p)$ and $\mathbf{d}_P(p) = \mathbf{L}_P(p,p)$. (2.19) indicates that λ_1 can be well approximated in linear time by finding the node with minimum weighted degree ratio of G and P.

Based on the above scalable methods for estimating the extreme eigenvalues λ_1 and λ_n of $\mathbf{L}_P^{+}\mathbf{L}_G$, as well as the weight sensitivity in (2.16), the following constrained nonlinear optimization framework for scaling up edge weights in P has been proposed.

$$
\begin{aligned}
&\textbf{minimize: } \lambda_n(w_P) \\
&\qquad\textbf{s. t.:} \\
&\textbf{(a) } \mathbf{L}_G\mathbf{u}_i = \lambda_i \mathbf{L}_P\mathbf{u}_i, i = 1, ..., n; \\
&\textbf{(b) } \lambda_{max} = \lambda_n \geq \lambda_{n-1} ... \geq \lambda_1 = \lambda_{min}; \\
&\textbf{(c) } \lambda_1^{(f)} \geq \lambda_1^{(0)} \overline{\Delta}_{\lambda_1}.
\end{aligned}
\tag{2.20}
$$

In the above formulation, $\lambda_1^{(0)}$ and $\lambda_1^{(f)}$ represent the smallest nonzero eigenvalue before and after subgraph scaling, respectively. $\overline{\Delta}_{\lambda_1}$ represents the upper bound of reduction factor in $\lambda_1^{(0)}$ after edge scaling. (2.20) aims to minimize λ_n by scaling up subgraph edge weights while limiting the decrease in λ_1.

A constrained SGD algorithm with momentum [96] has been proposed for iteratively

Algorithm 1 Edge Scaling via Constrained SGD Iterations

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

Initialize: $k = 1$, $\eta^{(1)} = \eta_{max}$, $\Delta_{\lambda_1} = \left(\overline{\Delta}_{\lambda_1}\right)^{\frac{1}{K_{max}}}$;
For each subgraph edge $(p, q) \in E_P$, initialize $\Delta w_P^{(1)}(p, q) = 0$;
while $\left(\frac{\boldsymbol{\Delta}\boldsymbol{\lambda}_{\mathbf{n}}^{(\mathbf{k})}}{\boldsymbol{\lambda}_{\mathbf{n}}^{(\mathbf{k})}} \geq \epsilon\right) \wedge (k \leq K_{max})$ **do**
  Compute approximate eigenvector $\mathbf{h}_t^{(k)}$ by (2.13);
  for each edge $(p, q) \in E_P$ **do**
    $s_{p,q}^{(k)} := -\lambda_n^{(k)} \left(\mathbf{e}_{p,q}^\top \mathbf{h}_t^{(k)}\right)^2$ by (2.16);
    $\Delta w_P^{(k+1)}(p, q) := \alpha \Delta w_P^{(k)}(p, q) - \eta^{(k)} s_{p,q}^{(k)}$;
    $\phi(p) := \frac{\mathbf{d}_G(p)}{\mathbf{d}_P(p) + \Delta w_P^{(k+1)}(p,q)}$;
    $\phi(q) := \frac{\mathbf{d}_G(q)}{\mathbf{d}_P(q) + \Delta w_P^{(k+1)}(p,q)}$;
    if $\min\left(\phi(p), \phi(q)\right) \leq \lambda_1^{(k)} \Delta_{\lambda_1}$ **then**
      $\Delta w_p := \frac{\mathbf{d}_G(p)}{\lambda_1^{(k)} \Delta_{\lambda_1}} - \mathbf{d}_P(p)$;
      $\Delta w_q := \frac{\mathbf{d}_G(q)}{\lambda_1^{(k)} \Delta_{\lambda_1}} - \mathbf{d}_P(q)$;
      $\Delta w_P^{(k+1)}(p, q) := \min\left(\Delta w_p, \Delta w_q\right)$;
    end if
    $w_P(p, q) := w_P(p, q) + \Delta w_P^{(k+1)}(p, q)$;
    $\mathbf{d}_P(p) := \mathbf{d}_P(p) + \Delta w_P^{(k+1)}(p, q)$;
    $\mathbf{d}_P(q) := \mathbf{d}_P(q) + \Delta w_P^{(k+1)}(p, q)$;
  end for
  $\eta^{(k+1)} := \frac{\lambda_n^{(k)}}{\lambda_n^{(0)}} \eta_{max}$;
  $k := k + 1$;
  Update $\lambda_1^{(k)}$ & $\lambda_n^{(k)}$ by (2.19);
  $\Delta\lambda_n^{(k)} := \lambda_n^{(k-1)} - \lambda_n^{(k)}$;
end while
Return the sparsified graph.

scaling up edge weights, as shown in Algorithm 1. The algorithm inputs include: the graph Laplacians $\mathbf{L}_G$ and $\mathbf{L}_P$, vectors $\mathbf{d}_G$ and $\mathbf{d}_P$ for storing diagonal elements in Laplacians, the initial largest and smallest generalized eigenvalues $\lambda_n^{(0)}$ and $\lambda_1^{(0)}$, the upper bound reduction factor $\overline{\Delta}_{\lambda_1}$ for λ_1, the coefficient α 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 K_{max}. **Lines 1-2** initialize parameters for the following SGD iterations. **Line 3** monitors the convergence condition for SGD iterations. **Lines 6-7** compute the weight update in SGD using the latest sensitivity and the previous update (momentum). **Lines 8-17** check the impact on λ_1 due to weight update: if λ_1 decreases significantly, an upper bound for weight update is applied; otherwise, directly apply the weight update computed in the previous steps.

2.2.4 Phase (C): Effective-Resistance Preserving Post Scaling

The last phase of our approach for spectral reduction of graphs is to globally scale up edge weights in the sparsified reduced graph to further improve the spectral approximation quality. Consider the following analysis for undirected graphs. Denote the non-decreasing 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. Then the following spectral decompositions of $\mathbf{L}_G$ and $\mathbf{L}_G^+$ always hold:

$$\mathbf{L}_G = \sum_{i=1}^{n-1} \zeta_i \omega_i \omega_i^\top, \qquad \mathbf{L}_G^+ = \sum_{j=1}^{n-1} \frac{1}{\zeta_j} \omega_j \omega_j^\top, \tag{2.21}$$

which leads to the following effective resistance between nodes p and q in G:

$$R_e^G(p,q) \;=\; e_{pq}^T \mathbf{L}_G^+ \mathbf{e}_{pq} = \sum_{i=1}^{n-1} \frac{1}{\zeta_i} \big(\mathbf{e}_{pq}^T \omega_i\big)^2. \tag{2.22}$$

(2.22) shows a close connection between the effective resistance metric and the first few Laplacian eigenvalues and eigenvectors. If we consider the graph as a resistor network with each conductance value corresponding to each edge weight, $R_e^G(p,q)$ can be regarded as the power dissipation of the resistor network when a unit current is flowing into node p and out from node q. By replacing the current vector $\mathbf{e}_{pq}$ with a random vector $\mathbf{b}^\perp \in R^n$ orthogonal to the all-one vector, it can be shown that matching the energy dissipated in the reduced graph and the original graph will immediately lead to improved approximation of the first few Laplacian eigenvalues and eigenvectors associated with the reduced graph.

It should be noted that the above scheme requires solving the original and reduced Laplacians once for finding the scaling factor, which can be achieved in almost-linear time leveraging graph-theoretic Laplacian solvers [64, 116].

2.2.5 Algorithm Complexity of The Proposed Spectral Graph Reduction Approach

Algorithm 2 Algorithm Flow for Spectral Graph Reduction

Input: Original graph Laplacian $\mathbf{L}_G$, user-defined reduction ratio ψ, graph density threshold $\gamma_{\max}$

1: Calculate graph density by $\gamma = \frac{|E_G|}{|V|}$;
2: **if** $\gamma < \gamma_{\max}$ **then**
3: Do node reduction (**Phase A**) on graph G to get graph R;
4: Apply spectral sparsification and edge scaling vis SGD (**Phase B**) on graph R to get graph S;
5: **else**
6: Apply spectral sparsification and edge scaling vis SGD (**Phase B**) on graph G to get graph P;
7: Do node reduction (**Phase A**) on graph P to get graph S;
8: **end if**
9: Performance post edge scaling scheme.
10: Return graph S and $\mathbf{L}_S$.

The complete algorithm flow for the proposed spectral graph reduction approach has been shown in Algorithm 2. The algorithm complexity of **Phase (A)** for the spectrum-preserving node reduction procedure is $O(|E_P|)$ for dense graphs and $O(|E_G|)$ for sparse graphs, the complexity of **Phase (B)** for spectral graph sparsification and edge scaling by SGD iterations is $O(|E_G| \log(|V|))$ for dense graphs and $O(|E_S| \log(|V_R|))$ for sparse graphs. Therefore, the worse-case algorithm complexity of the proposed spectral graph reduction method is $O(|E_G| \log(|V|))$. while the complexity of **Phase (C)** for post scaling is $O(|E_G|)$ when the recent graph-theoretic AMG solvers are leveraged [64, 116] for solving the original Laplacian matrix $\mathbf{L}_G$. Therefore, the worse-case algorithm complexity of the proposed spectral graph reduction method is $O(|E_G| \log(|V|))$.

2.2.6 Solution Refinement by Graph Filters

2.2.6.1 Graph Signal Processing and Spectral Sparsification/Reduction

To efficiently analyze signals on undirected graphs, graph signal processing techniques have been extensively studied in recent years [87]. There are analogies between traditional signal processing or classical Fourier analysis and graph signal processing [87]: (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. The spectrally sparsified/reduced graphs can be regarded as "low-pass" filtered graphs, which have retained as few as possible edges/nodes for preserving the slowly-varying or "low-frequency" signals on the original graphs. Consequently, spectrally sparsified/reduced graphs will be able to preserve the eigenvectors associated with low eigenvalues more accurately than high eigenvalues.

2.2.6.2 Solution Error due to Spectral Sparsification

Denote the non-decreasing eigenvalues and the corresponding unit-length, mutually-orthogonal eigenvectors of $\mathbf{L}_G$ by $0 = \zeta_1 < \zeta_2 \leq \cdots \leq \zeta_n$, and $\omega_{\mathbf{1}}, \cdots, \omega_{\mathbf{n}}$, respectively. Similarly denote the eigenvalues and eigenvectors of $\mathbf{L}_P$ by $0 = \tilde{\zeta}_1 < \tilde{\zeta}_2 \leq \cdots \leq \tilde{\zeta}_n$ and $\tilde{\omega}_{\mathbf{1}}, \cdots, \tilde{\omega}_{\mathbf{n}}$, respectively. It should be noted that both ω_1 and $\tilde{\omega}_1$ are the normalized all-one vector $\mathbf{1}/\sqrt{n}$. Then the following spectral decompositions of $\mathbf{L}_G$ and $\mathbf{L}_P$ will hold:

$$\mathbf{L}_G = \sum_{i=1}^{n} \zeta_i \omega_{\mathbf{i}} \omega_{\mathbf{i}}^{\top}, \mathbf{L}_P = \sum_{i=1}^{n} \tilde{\zeta}_i \tilde{\omega}_{\mathbf{i}} \tilde{\omega}_{\mathbf{i}}^{\top}. \tag{2.23}$$

We assume that the k smallest eigenvalues and their eigenvectors of $\mathbf{L}_G$ have been pretty well-preserved in $\mathbf{L}_P$, while the remaining $n-k$ eigenvalues and eigenvectors are not. Consequently the following approximate spectral decompositions of $\mathbf{L}_P$ will hold:

$$\mathbf{L}_P \approx \sum_{i=1}^{k} \zeta_i \omega_{\mathbf{i}} \omega_i^{\top} + \sum_{i=k+1}^{n} \tilde{\zeta}_i \tilde{\omega}_{\mathbf{i}} \tilde{\omega}_{\mathbf{i}}^{\top}. \tag{2.24}$$

In the following, we show that using spectrally-sparsified graphs for solving sparse matrix problems will only result in solution errors that can be expressed with eigenvectors associated with large eigenvalues, while the error analysis for spectrally-reduced graphs will be quite similar and omitted in this work. Consider the following SDD

matrix solution problem:

$$(\mathbf{L}_G + \delta\mathbf{I})\mathbf{x} = \mathbf{b}^{\perp}, \tag{2.25}$$

where $\mathbf{b}^{\perp} \in R^n$ is a random right-hand-side (RHS) vector orthogonal to the all-one vector $\mathbf{1}$, δ is a small positive real value added to graph Laplacian for modeling boundary conditions, and $\mathbf{I} \in R^{n\times n}$ is an identity matrix that can be written as follows:

$$\mathbf{I} = \sum_{i=1}^{n} \omega_{\mathbf{i}}\omega_{\mathbf{i}}^{\top} \approx \sum_{i=1}^{k} \omega_{\mathbf{i}}\omega_{i}^{\top} + \sum_{i=k+1}^{n} \tilde{\omega}_{\mathbf{i}}\tilde{\omega}_{\mathbf{i}}^{\top}, \tag{2.26}$$

we can rewrite $\mathbf{L}_G + \delta\mathbf{I}$ as follows:

$$\mathbf{L}_G + \delta\mathbf{I} = \sum_{i=1}^{n} (\zeta_i + \delta)\omega_{\mathbf{i}}\omega_{\mathbf{i}}^{\top}. \tag{2.27}$$

Consequently, $\mathbf{x}$ can be written as:

$$\mathbf{x} = \sum_{i=1}^{n} \frac{\omega_{\mathbf{i}}\omega_{\mathbf{i}}^{\top}\mathbf{b}^{\perp}}{\zeta_i + \delta}. \tag{2.28}$$

Let $\tilde{\mathbf{x}}$ denote the approximate solution obtained with $\mathbf{L}_P$:

$$\tilde{\mathbf{x}} \approx \sum_{i=k+1}^{n} \frac{\tilde{\omega}_{\mathbf{i}}\tilde{\omega}_{\mathbf{i}}^{\top}\mathbf{b}^{\perp}}{\tilde{\zeta}_i + \delta} + \sum_{i=1}^{k} \frac{\omega_{\mathbf{i}}\omega_{\mathbf{i}}^{\top}\mathbf{b}^{\perp}}{\zeta_i + \delta}, \tag{2.29}$$

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

$$\mathbf{e} = \mathbf{x} - \tilde{\mathbf{x}} \approx \sum_{i=k+1}^{n} \left(\frac{\omega_{\mathbf{i}} \omega_{\mathbf{i}}^{\top} \mathbf{b}^{\perp}}{\zeta_i + \delta} - \frac{\tilde{\omega}_{\mathbf{i}} \tilde{\omega}_{\mathbf{i}}^{\top} \mathbf{b}^{\perp}}{\tilde{\zeta}_i + \delta} \right). \tag{2.30}$$

(2.30) indicates that when using the sparsified graph Laplacian for solving the SDD matrix, the solution error can be expressed as a linear combination of eigenvectors corresponding to large Laplacian eigenvalues. Therefore, the error due to the sparsified graph Laplacian will be a combination of high frequency signals on graphs, which thus can be efficiently filtered out using "low-pass" graph signal filters [87].

2.2.6.3 Solution Refinement by Smoothing

Weighted-Jacobi or Gauss-Seidel methods, which have been widely adopted in modern iterative methods for solving large sparse matrices [83], such as the smoothing (relaxation) function in multigrid algorithms [64], can be applied for filtering out such high-frequency error signals on graphs. This work adopts a weighted-Jacobi iteration scheme for filtering eigenvectors on the graph, see Algorithm 3. The algorithm inputs include the original Laplacian matrix $\mathbf{L}_G$ that has been decomposed into a diagonal matrix $\mathbf{D}_G$ and an adjacency matrix $\mathbf{A}_G$, the approximate solution vectors obtained using sparsified Laplacian $\mathbf{L}_P$, as well as the weight ϑ and iteration number N_{iter} for signal filtering.

Algorithm 3 Solution Refinement Algorithm

Input: $\mathbf{L}_G = \mathbf{D}_G - \mathbf{A}_G$, $\tilde{\mathbf{x}}_\mathbf{1}$,..., $\tilde{\mathbf{x}}_\mathbf{k}$, ϑ, N_{iter};

1: For each of the approximate solution vectors $\tilde{\mathbf{x}}_\mathbf{1}$,..., $\tilde{\mathbf{x}}_\mathbf{k}$, do
2: **for** $i = 1$ **to** N_{iter} **do**
3: $\quad \tilde{\mathbf{x}}^{(\mathbf{i+1})} = (1 - \vartheta)\tilde{\mathbf{x}}^{(\mathbf{i})} + \vartheta \mathbf{D}_G^{-1}\mathbf{A}_G\tilde{\mathbf{x}}^{(\mathbf{i})}$
4: **end for**
5: Return the solution vectors $\tilde{\mathbf{x}}_\mathbf{1}$,..., $\tilde{\mathbf{x}}_\mathbf{k}$.

2.3 Spectral Reduction for Multilevel Graph Partitioning and Data Visualization

In this section, multilevel frameworks that leverages spectrally-reduced graphs for accelerated spectral graph partitioning as well as accelerated t-distributed Stochastic Neighbor Embedding (t-SNE) of large data sets are introduced. A more in-depth discussion of spectral partitioning and data clustering can be found in the Appendix.

2.3.1 Multilevel Laplacian Eigensolver for Scalable Spectral Graph Partitioning

The k-way spectral graph partitioning (clustering) algorithm is described in Algorithm 4 [85, 100]; the Laplacian eigensolver (line 2) is usually the computational bottleneck when working with large graphs. Here, we proposed a multilevel Laplacian

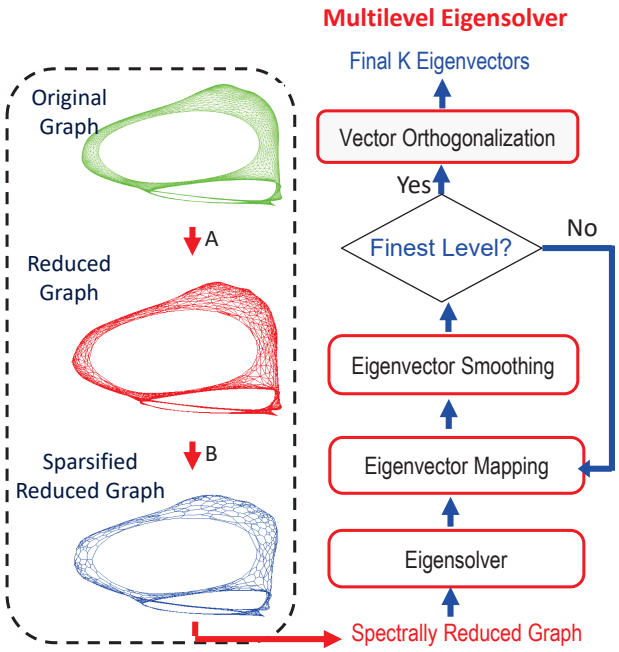


Figure 2.2: Multilevel Laplacian eigensolver for spectral graph partitioning.

Algorithm 4 K-Way Spectral Graph Partitioning

Input: Laplacian matrix $\mathbf{L}_G = \mathbf{D}_G - \mathbf{A}_G$, number of partitions k ;

1: Let $\mathbf{B_G} = \mathbf{I}$(ratio cut) or $\mathbf{B_G} = \mathbf{D_G}$ (normalized cut);
2: Compute the first k eigenvectors $\mathbf{u_1}, \cdots, \mathbf{u_k}$ of eigenproblem $\mathbf{L_G u_i} = \lambda_i \mathbf{B_G u_i}$ for $i = 1, \cdots, k$;
3: Form the matrix $\mathbf{U} \in \mathbb{R}^{nxk}$ with vectors $\mathbf{u_1}, \cdots, \mathbf{u_k}$ as columns;
4: Cluster the k-dimensional points defined by the rows of $\mathbf{U}$ with k-means algorithm;
5: Return partition $S_1, \cdots, S_k$;

eigensolver for more efficiently solving eigenvalue problems by leveraging spectrally-reduced graphs. Note that only the first few nontrivial eigenvectors of the original graph Laplacian are needed for spectral partitioning (clustering).

The algorithm flow of the proposed multilevel eigensolver is shown in Figure 2.2. Instead of directly computing the first k eigenvectors of the generalized eigenvalue

problem $\mathbf{L_G u_i} = \lambda_i \mathbf{B_G u_i}$, we will first reduce the original graph G into a much smaller graph S with the multilevel graph reduction scheme such that the eigenvectors of the reduced graph can be efficiently calculated. Next, we will map the eigenvectors of the reduced graph Laplacian onto a finer level using the graph mapping operators (as shown Table 2.1) determined during node aggregation procedure (**Phase A**). To further improve the approximation quality of these eigenvectors, we apply an eigenvector refinement (smoothing) procedure similar to Algorithm 3. The eigenvector mapping and smoothing procedures are recursively applied until the finest-level graph is reached. Finally, all eigenvectors for the finest-level graph will be orthonormalized using the Gram-Schmidt process.

The proposed eigenvector smoothing process essentially finds an approximate solution to the following equations for $i = 1, \cdots, k$:

$$(\mathbf{L}_\mathbf{G}^\upsilon - \lambda_i^\Upsilon \mathbf{B}_\mathbf{G}^\upsilon)\mathbf{u}_\mathbf{i}^\upsilon = 0, \tag{2.31}$$

where $\mathbf{L}_\mathbf{G}^\upsilon = \mathbf{D}_\mathbf{G}^\upsilon - \mathbf{A}_\mathbf{G}^\upsilon$ is the Laplacian on level υ after graph reduction, where $\upsilon = 1$ represents the finest level. We use Υ for denoting the coarsest (bottom) level, where $\mathbf{L}_G^\Upsilon = \mathbf{L}_S$; $\mathbf{B}_\mathbf{G}^\upsilon = \mathbf{I}$ will be used for ratio cut and $\mathbf{B}_\mathbf{G}^\upsilon = \mathbf{D}_\mathbf{G}^\upsilon$ for normalized cut (see Section A.1 in the Appendix for more details); λ_i^Υ is the eigenvalue of following generalized eigenproblem:

$$\mathbf{L}_G^{\Upsilon}\mathbf{u}_i^{\Upsilon} = \lambda_i^{\Upsilon}\mathbf{B}_G^{\Upsilon}\mathbf{u}_i^{\Upsilon} \tag{2.32}$$

Algorithm 5 Multilevel Laplacian Eigensolver

Input: $\mathbf{L}_G^1, \cdots, \mathbf{L}_G^{\Upsilon}, \quad \mathbf{H}_2^1, \cdots, \mathbf{H}_{\Upsilon}^{\Upsilon-1}, \quad k;$

1: Initialize: $j := \Upsilon$, $\mathbf{B}_{\mathbf{G}}^{\upsilon} := \mathbf{I}$ for ratio cut or $\mathbf{B}_{\mathbf{G}}^{\upsilon} := \mathbf{D}_{\mathbf{G}}^{\upsilon}$ for normalized cut, where $\upsilon = 1, \cdots, \Upsilon$;
2: Compute the first k eigenpairs $(\lambda_1^{\Upsilon}, \mathbf{u}_1^{\Upsilon}), \cdots, (\lambda_k^{\Upsilon}, \mathbf{u}_k^{\Upsilon})$ of the eigenvalue problem $\mathbf{L}_G^{\Upsilon}\mathbf{u}_i^{\Upsilon} = \lambda_i^{\Upsilon}\mathbf{B}_G^{\Upsilon}\mathbf{u}_i^{\Upsilon}$ for $i = 1, \cdots k$;
3: Form the matrix $\mathbf{U}^{\Upsilon}$ with vectors $\mathbf{u}_1^{\Upsilon}, \cdots, \mathbf{u}_k^{\Upsilon}$ as its columns;
4: **while** $j > 1$ **do**
5: Map $\mathbf{U}^j$ from level j to level $j-1$ by $\mathbf{U}^{j-1} = \mathbf{H}_j^{j-1}\mathbf{U}^j$;
6: **for** $i = 1$ **to** k **do**
7: $\mathbf{y} := \mathbf{U}^{j-1}[:, i]$, which is the i-th column of $\mathbf{U}^{j-1}$;
8: Filter vector $\mathbf{y}$ by performing a few weighted-Jacobi iterations to $(\mathbf{L}_G^{j-1} - \lambda_i^{\Upsilon}\mathbf{B}_G^{j-1})\mathbf{y} = 0$;
9: Update $\mathbf{U}^{j-1}[:, i]$ with the smoothed vector $\mathbf{y}$;
10: **end for**
11: $j := j - 1$;
12: **end while**
13: Perform orthonormalization to columns of $\mathbf{U}^1$;
14: Return $\mathbf{U} = \mathbf{U}^1$.

The detailed algorithm for multilevel Laplacian eigensolver is shown in Algorithm 5. The inputs of the algorithm include the Laplacian matrix of each hierarchical level $\mathbf{L}_G^{\upsilon} = \mathbf{D}_G^{\upsilon} - \mathbf{A}_G^{\upsilon}$, where $\upsilon = 1, \cdots, \Upsilon$; mapping operator $\mathbf{H}_{\upsilon}^{\upsilon-\mathbf{1}}$ from level υ to level $\upsilon - 1$; and the number of eigenvectors k. Spectral partitioning or clustering can be performed using the eigenvectors computed by Algorithm 5 in a subsequent k-means clustering step.

2.3.2 Multilevel t-SNE Algorithm for Scalable Data Visualization

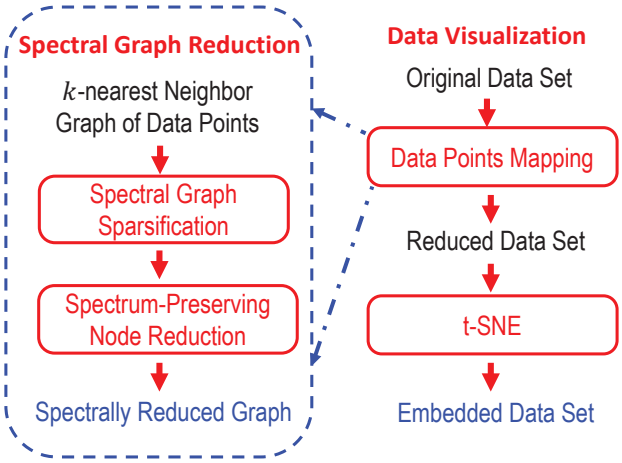


Figure 2.3: Multilevel t-SNE algorithm.

Visualization of high-dimensional data is a fundamental problem in data analysis and is used in many applications, such as medical sciences, physics, and economy. In recent years, the t-Distributed Stochastic Neighbor Embedding (t-SNE) has become an effective visualization tool with the capability to perform dimensionality reduction in such a way that the similar data points in high-dimensional space are embedded onto nearby locations in low-dimensional space of two or three dimensions with high probability. However, t-SNE may suffer from very high computational cost for visualizing large real-world data sets due to the superlinear computational complexity $O(N^2)$ [69, 99], where N is the number of data points in the data set.

Recent research shows that there is a clear connection between spectral graph partitioning (data clustering) and t-SNE [63]: the low-dimensional embedding obtained with t-SNE is closed related to the first few eigenvectors of the corresponding graph Laplacian that encodes the manifold of the original high-dimensional data points. This motivates us to leverage the spectrally-reduced graphs for computing similar t-SNE embedding results by proposing a multilevel t-SNE algorithm, as described in Algorithm 6 and shown in Figure 2.3.

The main idea of our multilevel t-SNE algorithm is to aggregate the data points that are closely related to each other on the manifold into much smaller sets, such that visualizing the reduced data set using t-SNE will be much faster and produce similar embedding results. To this end, we start by constructing a nearest-neighbor (NN) graph, such as the k-NN graph, for the original high-dimensional data points. Then a spectrally-reduced (NN) graph is computed using the proposed spectral reduction algorithm. Note that for k-NN graphs, the graph sparsification and scaling procedure (Phase B) will be performed before the spectral node aggregation step (Phase A).

Algorithm 6 Multilevel Data Visualization with t-SNE

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

1: Generate k-nearest neighbor (k-NN) graph G based on the data set F;
2: Generate the spectrally-reduced graph S;
3: Form the mapping operators such that $\mathbf{L}_S = \mathbf{H}_G^R \mathbf{L}_G \mathbf{H}_R^G$;
4: Form a reduced data set $\mathbf{F}_R$ by $\mathbf{F}_R = \mathbf{H}_G^R \mathbf{F}$;
5: Embed data points with t-SNE on the reduced data set $\mathbf{F}_R$;
6: Return embedded data points for visualization.

2.4 Experimental Results

In this section, extensive experiments have been conducted to evaluate the proposed spectral graph reduction and its applications to spectral partitioning, hypergraph partitioning and data visualization with various types of graphs from the DIMACS10 graph collection [3, 4] and hypergraphs from ISPD98 circuit partitioning benchmark suite [1]. The graphs span various applications, such as finite-element analysis problems ("fe_tooth", "fe_rotor") [17], numerical simulation graphs ("wing_nodal"), clustering graphs ("uk") and social network graphs ("coAuthorsDBLP" and "coPapersCiterseer") [17], etc. All experiments were conducted on a single CPU core of a computing platform running 64-bit RHEL 6.0 with 2.67GHz 12-core CPU and 48GB DRAM memory.

2.4.1 Results of Spectrum Preservation on Spectrally Reduced Graphs

Figure 2.4 shows the spectral drawings [49] of the fe_ocean graph and its reduced graph computed by the proposed spectral graph reduction algorithm using the first two bottom eigenvectors, where the node and edge reduction ratio are $24X$ and $58X$, respectively. We observe that the spectral drawings of two graphs are highly similar to

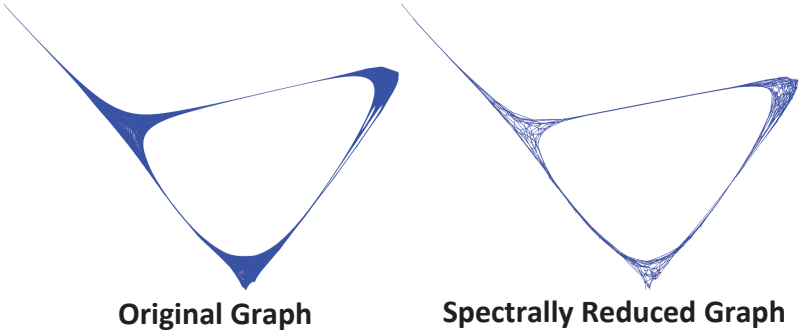


Figure 2.4: Spectral drawings of the "fe_ocean" graph and its reduced graph ($24X$ node reduction and $58X$ edge reduction).

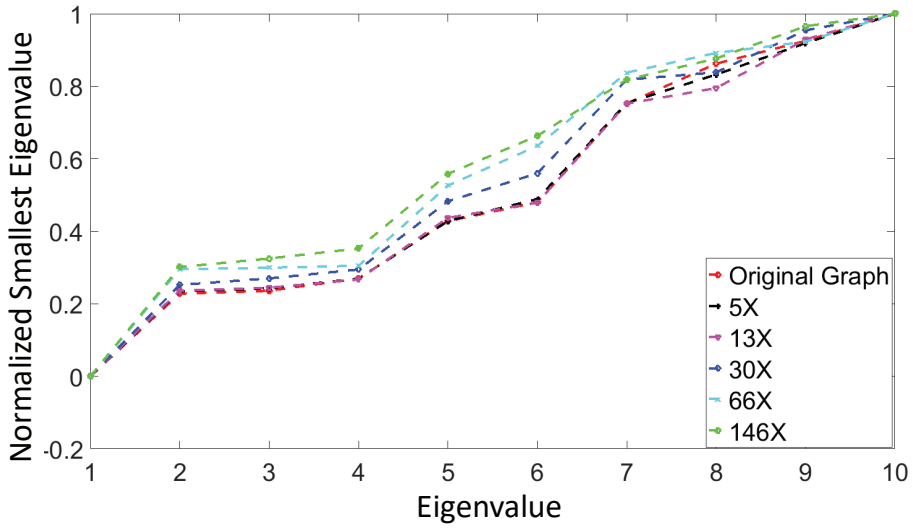


Figure 2.5: The first 10 normalized eigenvalues of the "fe_tooth" graph under different node reduction ratios.

each other, which indicates well-preserved spectral properties (Laplacian eigenvectors) in the reduced graph. Figure 2.5 shows the first few normalized eigenvalues of the original and reduced graph Laplacians, indicating clearly that the smallest eigenvalues of the original Laplacian and the reduced Laplacian match to each other even for very large reduction ratios.

Table 2.2
Mean relative errors for the first 10 and 40 eigenvalues.

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

Table 2.3
Edge number for reduced graphs using different reduction methods.

Graph	loca. (ed)	loca. (ng)	heav.	Kron	ours
airfoil	3126	3246	3322	589487	1049
yeast	713	779	603	60806	390
bunny	8897	11059	8838	280875	981
minnesota	1264	1259	603	3675	732

We also compared the performance of our proposed method with the following state-of-the-art graph coarsening methods:

(1) local variation based graph coarsening method [66, 67]. Based on the concept of restricted spectral approximation, two possible graph contraction methods were proposed: edge-based contraction (noted as loca. (ed) in the tables) and neighborhood-based contraction (noted as loca. (ng) in the tables).

(2) heavy edge matching based graph coarsening method (noted as heav. in the tables), which is widely used for graph partitioning [42] and more recently in graph embedding [61].

(3) Kron reduction method [86]. The benefit of this method is that it can preserve the important spectral properties; however, the densities of reduced graphs will be dramatically increased.

To measure the performance of different spectral coarsening methods, the mean relative eigenvalue errors between original graphs and reduced graphs are reported in Table 2.2, where five methods are tested, including local variation with edge and neighborhood contraction, heavy edge contraction, Kron reduction, as well as our proposed coarsening method; r represents the reduction ratio, which can be calculated by $1 - |V_S|/|V|$; $|V|$ and $|V_S|$ are the number of node for the original graph and the reduced graph, respectively. Given the first k eigenvalues ζ_i and $\tilde{\zeta}_i$ of the original graph and the reduced graph, the mean relative error can be calculated by $\frac{1}{k}\sum_{i=1}^{k}\frac{|\zeta_i-\tilde{\zeta}_i|}{\zeta_i}$ [65]. Four different graphs including airfoil ($|V| = 4000$, $|E_G| = 11490$) [79], yeast ($|V| = 1458$, $|E_G| = 1948$) [40], bunny ($|V| = 2503$, $|E_G| = 65490$) [98] and Minnesota ($|V| = 2642$, $|E_G| = 3304$) are tested in the experiment. We can observe that the spectrum can be better preserved on the reduced graphs using our proposed graph coarsening algorithm compared to other methods. Table 2.3 shows the number of the edges for the reduced graphs when using the different reduction methods. We can observe that our method can achieve better graph sparsity when comparing to other methods.

Table 2.4
Results of Effective-Resistance Preserving Spectral Graph Reduction.

Test Cases	Original Graph (G)		Reduced Graph w/o Sparsification (R)				Reduced Graph w/ Sparsification (S)				Total time
	$\|V\|$	$\|E_G\|$	$\|V_R\|$ $\left(\frac{\|V\|}{\|V_R\|}\right)$	$\|E_R\|$ $\left(\frac{\|E_G\|}{\|E_R\|}\right)$	$Er_R(\%)$	$T_r(s)$	$\|V_R\|$ $\left(\frac{\|V\|}{\|V_R\|}\right)$	$\|E_S\|$ $\left(\frac{\|E_G\|}{\|E_S\|}\right)$	$Er_S(\%)$	$T_s(s)$	$T_{tot}(s)$
fe_rotor	$1.0E5$	$6.6E5$	$3.4E3$ $(29X)$	$2.6E4$ $(25X)$	2.7%	$1.26s$	$3.4E3$ $(29X)$	$6.9E3$ $(95X)$	2.4%	$3.54s$	$4.80s$
fe_ocean	$1.4E5$	$4.1E5$	$6.1E3$ $(24X)$	$3.2E4$ $(13X)$	7.7%	$1.02s$	$6.1E3$ $(24X)$	$7.0E3$ $(58X)$	6.0%	$3.65s$	$4.67s$
parabolic_fem	$5.3E5$	$1.6E6$	$1.1E4$ $(46X)$	$3.4E4$ $(47X)$	7.9%	$3.94s$	$1.1E4$ $(46X)$	$1.2E4$ $(128X)$	7.8%	$13.59s$	$17.53s$
2D_mesh	$4.0E4$	$8.0E4$	$4.3E3$ $(9X)$	$1.2E4$ $(7X)$	8.3%	$0.26s$	$4.3E3$ $(9X)$	$4.7E3$ $(17X)$	7.2%	$1.59s$	$1.85s$
3D_thermal	$4.8E5$	$1.4E6$	$7.9E3$ $(61X)$	$5.4E4$ $(26X)$	5.2%	$3.17s$	$7.9E3$ $(61X)$	$8.7E3$ $(163X)$	4.4%	$11.18s$	$13.35s$
3D_laplacian	$1.0E6$	$3.0E6$	$8.1E3$ $(124X)$	$5.5E4$ $(54X)$	5.0%	$7.38s$	$8.1E3$ $(124X)$	$8.9E3$ $(334X)$	4.3%	$18.66s$	$26.04s$
Gmat_thu1	$5.0E6$	$8.2E6$	$9.7E4$ $(51X)$	$2.9E5$ $(29X)$	9.2%	$26.52s$	$9.7E4$ $(51X)$	$1.1E5$ $(78X)$	6.2%	$214.75s$	$241.27s$
Gmat_airfoil	$4.2E3$	$1.2E4$	$8.3E2$ $(5X)$	$2.3E3$ $(5X)$	7.8%	$0.07s$	$8.3E2$ $(5X)$	$9.6E2$ $(13X)$	6.3%	$0.08s$	$0.15s$
ecology	$1.0E6$	$2.0E6$	$1.1E5$ $(9X)$	$2.9E5$ $(7X)$	9.4%	$6.05s$	$1.1E5$ $(9X)$	$1.2E5$ $(17X)$	6.8%	$28.89s$	$34.94s$
*appu**	$1.4E4$	$9.2E5$	$1.3E2$ $(107X)$	$7.0E3$ $(131X)$	0.1%	$9.53s$	$1.3E2$ $(107X)$	$1.3E2$ $(7,019X)$	0.1%	$0.03s$	$9.56s$
*vsp_msc**	$2.2E4$	$1.2E6$	$2.2E2$ $(100X)$	$4.4E3$ $(280X)$	5.6%	$11.16s$	$2.2E2$ $(100X)$	$2.3E2$ $(5,427X)$	4.8%	$0.03s$	$11.19s$
auto	$4.5E5$	$3.3E6$	$2.9E3$ $(153X)$	$2.1E4$ $(157X)$	4.2%	$5.71s$	$2.9E3$ $(153X)$	$3.1E3$ $(1,079X)$	4.2%	$11.71s$	$17.41s$
coAuthorsDBLP	$3.0E5$	$9.7E5$	$1.3E3$ $(233X)$	$5.5E4$ $(18X)$	3.3%	$2.36s$	$1.3E3$ $(233X)$	$1.6E3$ $(603X)$	3.2%	$10.05s$	$12.41s$
coPapersDBLP	$5.4E5$	$1.5E7$	$1.6E3$ $(347X)$	$9.9E4$ $(154X)$	4.6%	$12.70s$	$1.6E3$ $(347X)$	$1.6E3$ $(9,336X)$	4.6%	$21.25s$	$33.95s$
coPapersCiteseer	$4.3E5$	$1.6E7$	$5.3E2$ $(816X)$	$2.1E4$ $(748X)$	4.0%	$10.07s$	$5.3E2$ $(816X)$	$5.6E2$ $(28,843X)$	3.6%	$16.69s$	$26.76s$

2.4.2 Results of Effective-Resistance Preservation on Spectrally Reduced Graphs

Table 2.4 shows graph reduction results on different graphs using the proposed method, where effective resistances errors are reported. Er_R (Er_S) denotes the average relative errors of effective resistances between graph R (S) and graph G; T_r, T_s and T_{tot} denote the spectral reduction time, spectral graph sparsification with edge scaling time, and total reduction runtime, respectively. The relative errors of effective resistance values are computed by averaging the relative errors of the effective resistances computed for 100 randomly selected node pairs.

Compared to other test cases that correspond to sparse graphs, the graphs "*appu**" and "*vsp_msc**" have much higher densities and thus been processed as dense graphs. We want to emphasize that directly applying the prior algebraic-distance-based node

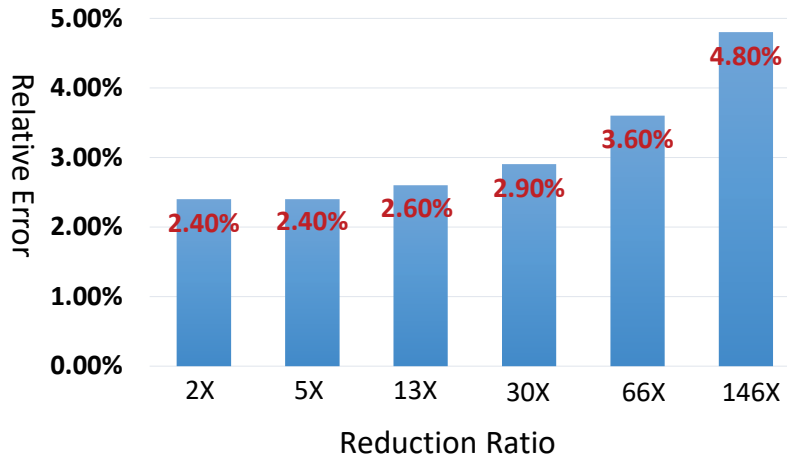


Figure 2.6: Average relative errors of effective resistance under different graph reduction ratios for the "fe_tooth" graph.

aggregation scheme [10] for node reduction may not always produce acceptable results for dense graphs. For example, the node aggregation algorithm failed to generate the reduced graph for "*appu**" due to its high graph density. On the other hand, there will be no issue when **Phase (B)** for spectral graph sparsification and scaling is applied before the node aggregation phase. For all test cases, it is observed that the effective resistances computed with the original graphs can be well approximated by using the spectrally reduced graphs (R) and sparsified reduced graphs (S).

The average relative errors of 100 randomly computed effective resistances have been shown in Figure 2.6 with different spectral graph reduction ratios for the "fe_tooth" graph. We observe that the effective resistance accuracy will drop slightly when higher reduction ratios are used.

Figure 2.7 shows the total spectral graph reduction time with different problem sizes

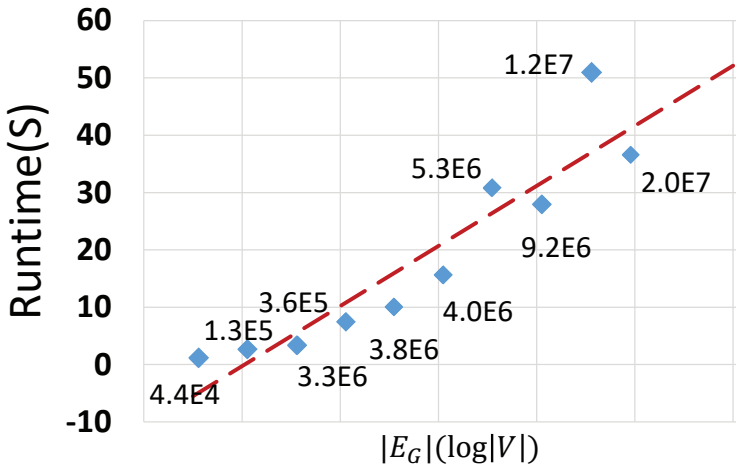


Figure 2.7: Runtime scalability of proposed spectral graph reduction method.

($|E_G|\log(|V|)$) for various graphs, where $|E_G|$ ($|V|$) denotes the number of edges (nodes) of the original graphs, respectively. As observed, the total spectral reduction runtime increases nearly-linearly with the problem size, indicating highly scalable performance of the proposed method ($O\left(|E_G|\log(|V|)\right)$).

2.4.3 Results of Scalable Spectral Graph Clustering (Partitioning)

We evaluated the performance of the proposed spectral graph partitioning algorithm on a varieties of graphs from the DIMACS10 graph collection. We choose to partition all the graphs into 30 partitions. The built-in `eigs` and `kmeans` MATLAB functions are used for solving the eigenvalue problem and node clustering tasks, respectively.

Table 2.5
Spectral Graph Reduction Results on Sample Graphs.

Test cases			Original Graph (G)		Spectrally Reduced Graph (S)		
Index	Graph	Application	$\|V\|$	$\|E_G\|$	$\|V_S\|$ $\left(\frac{\|V\|}{\|V_S\|}\right)$	$\|E_S\|$ $\left(\frac{\|E_G\|}{\|E_S\|}\right)$	$T_{reduction}$
1	fe_rotor	Finite Element	$1.0E5$	$6.6E5$	$1.4E3$ $(71X)$	$3.7E3$ $(180X)$	$1.30s$
2	fe_tooth	Finite Element	$7.8E4$	$4.5E5$	$1.3E3$ $(61X)$	$2.8E3$ $(162X)$	$0.94s$
3	auto	Numerical simulation	$4.5E5$	$3.3E6$	$1.5E4$ $(30X)$	$2.0E4$ $(167X)$	$14.81s$
4	wing_nodal	Numerical simulation	$1.1E4$	$7.5E4$	$1.8E2$ $(61X)$	$3.8E2$ $(197X)$	$0.21s$
5	luxembourg_osm	Street Network	$1.1E5$	$1.2E5$	$2.6E3$ $(44X)$	$3.2E3$ $(38X)$	$0.86s$
6	mi2010	US Census	$3.3E5$	$7.9E5$	$1.3E4$ $(26X)$	$1.6E4$ $(49X)$	$2.94s$
7	uk	Clustering	$4.8E3$	$6.8E3$	$1.2E2$ $(40X)$	$1.3E2$ $(51X)$	$0.22s$
8	smallworld	Clustering	$1.0E5$	$5.0E5$	$8.2E3$ $(12X)$	$2.1E4$ $(24X)$	$32.20s$
9	vsp_barth5_1Kse	Star Mixtures	$3.2E4$	$1.0E5$	$5.6E2$ $(57X)$	$8.3E2$ $(122X)$	$0.46s$
10	vsp_befref_fxm	Star Mixtures	$1.4E4$	$9.8E4$	$2.8E2$ $(49X)$	$8.1E3$ $(12X)$	$0.24s$
11	vsp_bump2_e18	Star Mixtures	$5.6E4$	$3.0E5$	$3.9E3$ $(14X)$	$1.3E5$ $(2.3X)$	$0.91s$
12	vsp_p0291_seymourl	Star Mixtures	$1.0E4$	$5.4E4$	$2.0E3$ $(5X)$	$5.1E3$ $(11X)$	$0.67s$
13	vsp_model1_crew1	Star Mixtures	$4.5E4$	$1.9E5$	$2.1E3$ $(21X)$	$4.6E3$ $(41X)$	$0.70s$
14	vsp_vibrobox_scagr7	Star Mixtures	$7.7E4$	$4.4E5$	$3.3E3$ $(23X)$	$9.4E3$ $(47X)$	$2.65s$
15	vsp_bcsstk30_500sep	Star Mixtures	$5.8E4$	$2.0E6$	$1.7E3$ $(34X)$	$3.1E3$ $(654X)$	$2.26s$
16	coAuthorsDBLP	Citations	$3.0E5$	$9.8E5$	$2.7E4$ $(11X)$	$3.8E4$ $(26X)$	$30.71s$
17	coAuthorsCiteseer	Citations	$2.2E5$	$8.1E5$	$2.0E4$ $(11X)$	$2.5E4$ $(33X)$	$8.20s$
18	citationCiteseer	Citations	$2.6E5$	$1.1E6$	$2.0E4$ $(13X)$	$4.1E4$ $(27X)$	$32.32s$
19	coPapersDBLP	Citations	$5.4E5$	$1.5E7$	$4.1E4$ $(13X)$	$7.3E4$ $(210X)$	$52.83s$
20	*coPapersCiteseer**	Citations	$4.3E5$	$1.6E7$	$1.3E4$ $(32X)$	$1.7E4$ $(950X)$	$16.41s$
21	*appu**	Random Graph	$1.4E4$	$9.2E5$	$2.8E3$ $(5X)$	$6.7E5$ $(1.4X)$	$25.53s$

The normalized cut (see Appendix A.1) is used to measure the quality of partitions. Even though the ratio cut and normalized cut are similar, they are trying to solve slightly different optimization problems, and one might be preferable over the other depending on the application. Two partitioning algorithms have been tested, including spectral partitioning with original graphs (no reduction) and spectral partitioning with graph reduction.

Table 2.5 shows spectral graph reduction results on different kinds of graphs using the proposed method, where $T_{reduction}$ denotes the spectral graph reduction time. Compared to other test cases that correspond to sparse graphs, the graph densities of "*coPapersCiteseer**" and "*appu**" are much higher and thus have been processed as dense graphs. We want to further emphasize that directly applying the prior

Table 2.6
Results of Graph Partitioning.

Test cases		Original Graph (G)			Spectrally Reduced Graph (S)			
Index	Graph	θ	T_{eigs}	T	θ	T_{eigs}	T_{smooth}	T
1	fe_rotor	1.51	$20.2s$	$22.8s$	1.50	$0.2s$	$2.9s$	$5.4s$
2	fe_tooth	1.77	$14.6s$	$16.6s$	1.68	$0.2s$	$1.8s$	$4.0s$
3	auto	1.10	$479.7s$	$495.8s$	1.08	$0.6s$	$12.3s$	$29.0s$
4	wing_nodal	4.88	$2.3s$	$3.3s$	4.71	$0.1s$	$0.4s$	$1.5s$
5	luxembourg_osm	0.07	$3.5s$	$6.3s$	0.07	$0.2s$	$0.9s$	$3.8s$
6	mi2010	0.43	$14.5s$	$21.6s$	0.41	$0.4s$	$3.7s$	$10.2s$
7	uk	1.03	$0.2s$	$0.6s$	1.05	$0.1s$	$0.1s$	$0.6s$
8	smallworld	7.02	$16,137.9s$	$16,144.5s$	7.05	$9.2s$	$2.8s$	$14.1s$
9	vsp_barth5_1Kse	3.12	$14.4s$	$16.6s$	2.72	$0.2s$	$0.5s$	$2.7s$
10	vsp_befref_fxm	13.59	$3.4s$	$4.7s$	12.83	$0.1s$	$0.4s$	$1.8s$
11	vsp_bump2_e18	14.60	$123.0s$	$124.7s$	13.55	$1.7s$	$1.4s$	$5.4s$
12	vsp_p0291_seymourl	8.09	$2.2s$	$2.9s$	7.88	$0.4s$	$0.2s$	$1.3s$
13	vsp_model1_crew1	11.38	$11.5s$	$13.9s$	10.48	$0.7s$	$0.8s$	$4.9s$
14	vsp_vibrobox_scagr7	6.92	$73.8s$	$75.8s$	6.85	$0.6s$	$2.3s$	$4.8s$
15	vsp_bcsstk30_500sep	†	†	†	2.09	$0.2s$	$24.0s$	$25.7s$
16	coAuthorsDBLP	0.92	$245.3s$	$250.8s$	0.49	$15.7s$	$4.2s$	$26.5s$
17	coAuthorsCiteseer	0.67	$77.0s$	$81.3s$	0.41	$5.4s$	$3.2s$	$13.3s$
18	citationCiteseer	0.48	$2,005.2s$	$2,027.7s$	0.52	$12.9s$	$4.9s$	$24.8s$
19	coPapersDBLP	NA	NA	NA	0.14	$17.4s$	$43.1s$	$61.6s$
20	coPapersCiteseer	NA	NA	NA	0.06	$0.87s$	$44.0s$	$51.6s$
21	appu*	22.47	$178.9s$	$179.9s$	23.80	$7.3s$	$3.4s$	$11.7s$

algebraic-distance-based node aggregation scheme [10] will not produce acceptable results. For example, the node aggregation algorithm failed to generate the reduced graph for "*appu**" due to very high graph density. On the other hand, there will be no issue for dense graphs if we apply **step B** for spectral graph sparsification and scaling before the node aggregation step.

The performance of partitioning is evaluated based on the normalized cut and total execution time. Detailed results have been shown in Table 2.6, where θ is the normalized cut, T_{eigs} is the execution time for solving the eigenvalue problem, T_{smooth} denotes eigenvector refinement (smoothing) time, T denotes the total runtime for

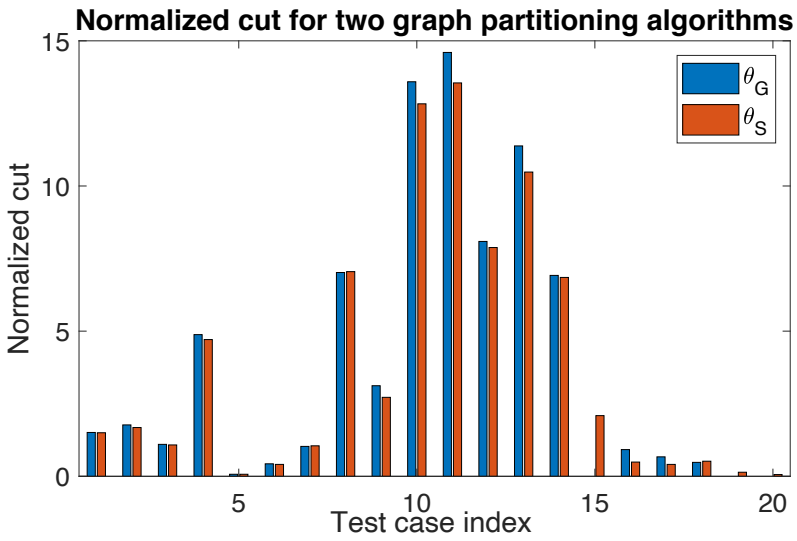


Figure 2.8: Normalized cut (partitioning quality) for spectral partitioning with the original graphs and reduced graphs.

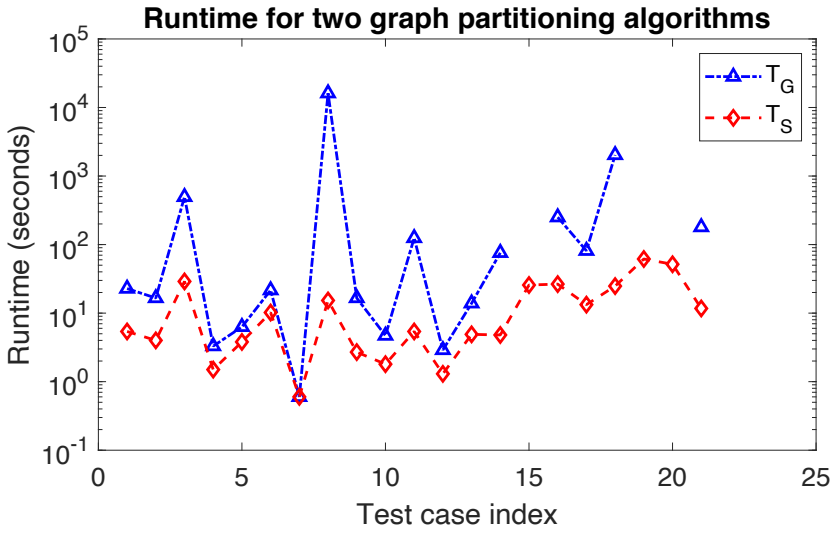


Figure 2.9: Execution time for graph partitioning when using the original graphs and spectrally reduced graphs.

spectral graph partitioning, † represents the failure of solving eigenvalue problems due to the singularity of the Laplacian matrix, and "NA" denotes the failure of solving eigenvalue problems due to the limited memory resources. To better compare

the performance of the two algorithms, we plot the clustering quality (normalized cut) when using spectral clustering with the original graph and the reduced graph in Figure 2.8, where smaller value of normalized cut represents better clustering quality. Meanwhile, the total execution times required by two graph clustering algorithms have also been shown in Figure 2.9, where T_G and T_S are the total partitioning time when using the original graph and the reduced graph. From the table and figures, we can observe that the overall quality of generated clusters by spectral clustering using the original graph and the coarsened graph is similar to each other, but the cost when using coarsened graph is much lower than using the original graph, especially for large graphs. For example, we achieve over $1100X$ runtime speedup on the "smallworld" graph. For larger graphs, such as the "coPapersCiteseer" graphs, spectral clustering without reduction will fail due to the extremely high computation (memory) cost.

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

Figure 2.10 shows the profiling of time required in spectral partitioning of the "auto" graph. It indicates that most of the runtime is due to the eigensolver if the original

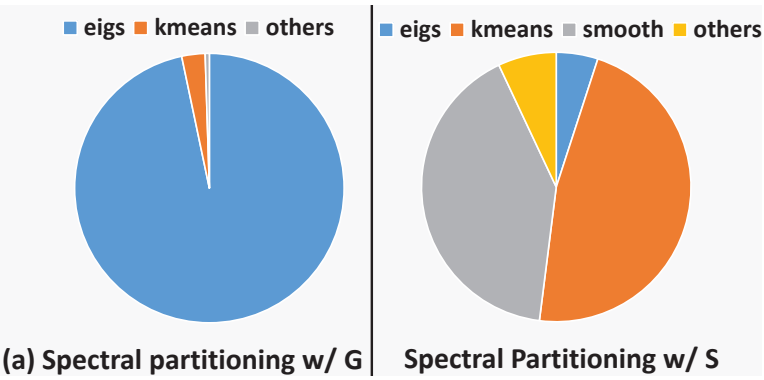


Figure 2.10: Profiling of time spent in spectral partitioning on "auto" graph [17].

graph is used, while the k-means and smoothing time will be dominant when using the spectrally-reduced graph. However, the smoothing procedure is inherently highly parallel making it possible to further improve the efficiency of the proposed spectral partitioning and to develop high-quality parallel spectral partitioning algorithms.

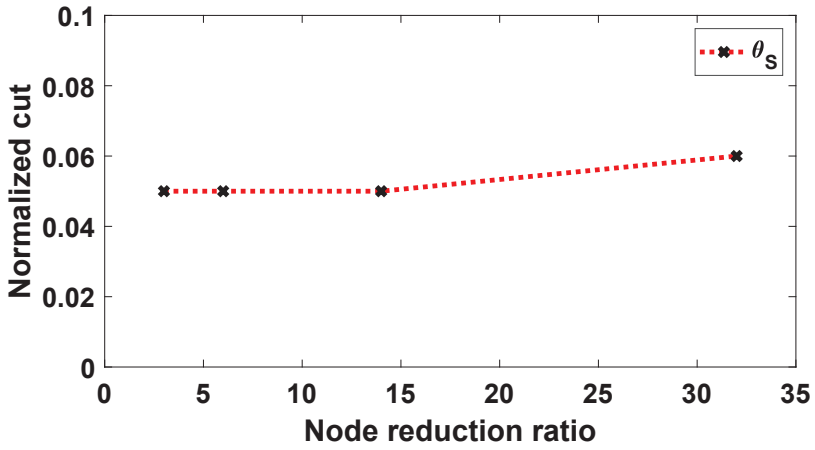


Figure 2.11: Partitioning qualities (normalized cut) under different reduction ratio for the "coPapersCiteseer" graph [17].

We also evaluated the performance of the proposed spectral partitioning method using different reduction ratios, as shown in Figure 2.11 and Figure 2.12. We observe that

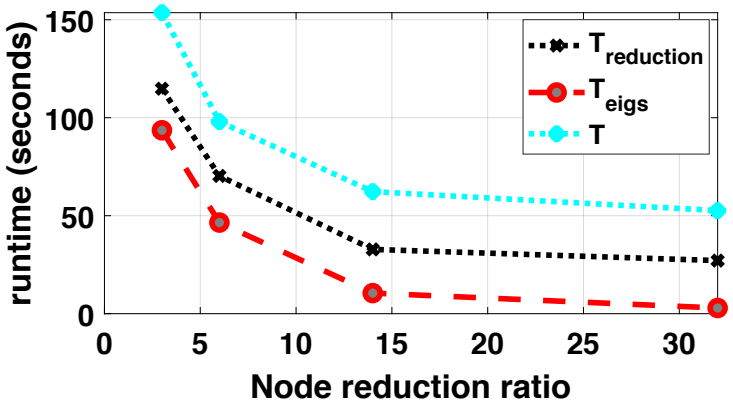


Figure 2.12: Runtime for multi-way spectral partitioning under different reduction ratio for the "coPapersCiteseer" graph [17].

higher graph reduction ratios immediately result in lower cost for graph reduction as well as spectral partitioning while still maintaining high partitioning quality. This indicates very promising performance in efficiency and reliability achieved by the proposed algorithm.

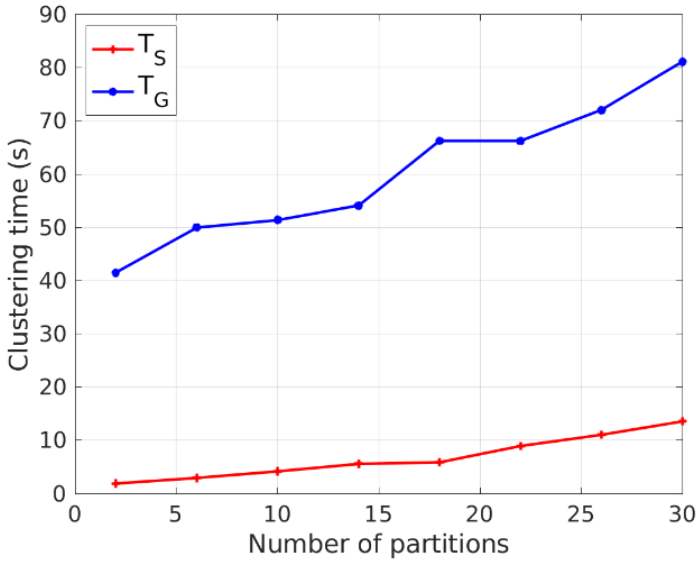


Figure 2.13: Runtime for graph partitioning with different clusters (partitions) for the "coAuthorsCiteseer" graph [17].

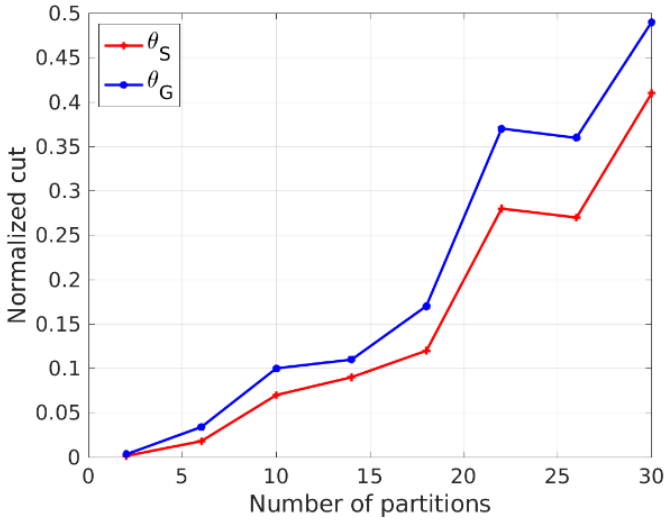


Figure 2.14: Normalized cut for graph partitioning with different clusters (partitions) for the "coAuthorsCiteseer" graph [17].

Finally, we evaluate the performance of two partitioning algorithms using different numbers of partitions. As shown in Figure 2.13 and Figure 2.14, the reduced graph has 11× fewer nodes and 26× fewer edges compare to the original graph. With the increasing number of partitions, we observed that the spectral partitioning method using the spectrally-reduced graph is much faster with consistently higher partitioning qualities.

2.4.4 Results of Hypergraph Partitioning

One approach to spectral hypergraph partitioning is to construct an undirected graph from the hypergraph and then apply spectral graph partitioning algorithms to the generated graph [41]. Existing methods for constructing an undirected graph from

Table 2.7
Benchmarks of Spectral Hypergraph Partitioning.

Benchmarks	Hypergraph (Hy)		Original Graph (G)		Spectrally Reduced Graph (S)		
	$\|V\|$	$\|E_H\|$	$\|V\|$	$\|E_G\|$	$\|V_S\|$ $\left(\frac{\|V\|}{\|V_S\|}\right)$	$\|E_S\|$ $\left(\frac{\|E_G\|}{\|E_S\|}\right)$	$T_r(s)$
ibm01	$1.3E4$	$1.4E4$	$1.3E4$	$1.1E5$	$1.0E3$ $(13X)$	$1.7E3$ $(63X)$	$0.39s$
ibm02	$2.0E4$	$2.0E4$	$2.0E4$	$3.4E5$	$1.6E3$ $(12X)$	$3.6E3$ $(97X)$	$0.98s$
ibm03	$2.3E4$	$2.7E4$	$2.3E4$	$2.1E5$	$4.6E3$ $(5X)$	$6.0E3$ $(35X)$	$0.95s$
ibm04	$2.7E4$	$3.2E4$	$2.7E4$	$2.2E5$	$5.4E3$ $(5X)$	$5.5E3$ $(40X)$	$1.17s$
ibm05	$2.9E4$	$2.8E4$	$2.9E4$	$3.5E5$	$5.6E3$ $(5X)$	$1.7E4$ $(21X)$	$5.54s$
ibm06	$3.2E4$	$3.5E4$	$3.2E4$	$3.2E5$	$2.6E3$ $(12X)$	$6.3E3$ $(51X)$	$2.18s$
ibm07	$4.6E4$	$4.8E4$	$4.6E4$	$3.7E5$	$3.7E3$ $(13X)$	$5.4E3$ $(69X)$	$1.49s$
ibm08	$5.1E4$	$5.1E4$	$5.1E4$	$7.3E5$	$4.8E3$ $(11X)$	$1.4E4$ $(53X)$	$3.48s$
ibm09	$5.3E4$	$6.1E4$	$5.3E4$	$4.8E5$	$4.6E3$ $(13X)$	$1.3E4$ $(38X)$	$2.68s$
ibm10	$6.9E4$	$7.5E4$	$6.9E4$	$7.1E5$	$5.9E3$ $(12X)$	$9.0E3$ $(79X)$	$2.66s$
ibm11	$7.1E4$	$8.1E4$	$7.1E4$	$5.1E5$	$5.7E3$ $(12X)$	$1.3E4$ $(39X)$	$3.68s$
ibm12	$7.1E4$	$7.7E4$	$7.1E4$	$7.5E5$	$1.5E4$ $(5X)$	$1.8E4$ $(41X)$	$4.09s$
ibm13	$8.4E4$	$1.0E5$	$8.4E4$	$7.4E5$	$7.8E3$ $(11X)$	$1.0E4$ $(76X)$	$3.04s$
ibm14	$1.5E5$	$1.5E5$	$1.5E5$	$1.1E6$	$1.1E4$ $(14X)$	$1.3E4$ $(83X)$	$4.41s$
ibm15	$1.6E5$	$1.9E5$	$1.6E5$	$1.8E6$	$1.4E4$ $(11X)$	$1.8E4$ $(99X)$	$4.53s$
ibm16	$1.8E5$	$1.9E5$	$1.8E5$	$1.9E6$	$1.5E4$ $(12X)$	$1.9E4$ $(101X)$	$6.91s$
ibm17	$1.9E5$	$1.9E5$	$1.9E5$	$2.2E6$	$1.6E4$ $(11X)$	$2.4E4$ $(95X)$	$19.71s$
ibm18	$2.1E5$	$2.0E5$	$2.1E5$	$2.2E6$	$1.9E4$ $(11X)$	$2.1E4$ $(107X)$	$8.54s$

a hypergraph are based on clique or star expansions [41, 44]. In this work, the clique expansion method is adopted by replacing each hyperedge with a complete subgraph for all vertices in that hyperedge. However, the size of the generated graph (both nodes and edges) can be greatly increased, which will introduce a very high cost when computing eigenvectors for spectral partitioning. To achieve good efficiency, the multilevel Laplacian eigensolver described in Algorithm 5 can be utilized to accelerate eigenvalue problems by leveraging spectrally-reduced graphs without loss of solution quality.

The performance of spectral hypergraph partitioning is evaluated on 18 hypergraphs from ISPD98 circuit partitioning benchmark suite [1], where only unit cell-areas are

Table 2.8
Performance of Spectral Hypergraph Partitioning on Original Graphs G.

ibm	Partitioning w/ Graph (G)					
	8-way		16-way		32-way	
	Cut (SD)	T_e^G	Cut (SD)	T_e^G	Cut (SD)	T_e^G
01	800 (1633)	1.35s	943 (1949)	1.43s	1461 (3057)	1.94s
02	1096 (2302)	4.44s	2039 (4244)	5.70s	3163 (6695)	8.27s
03	1532 (3089)	6.29s	2710 (5542)	7.26s	4419 (9166)	11.53s
04	1995 (4108)	5.40s	3127 (6614)	7.43s	4727 (9942)	10.58s
05	3100 (6263)	25.69s	4472 (9208)	53.86s	5289 (10865)	63.12s
06	2390 (4960)	12.10s	3367 (7193)	15.90	4507 (9611)	20.88s
07	1724 (3508)	13.71s	3785 (7666)	18.72s	6713 (13968)	23.36s
08	3343 (6815)	32.35s	4750 (9848)	33.69s	6318 (13117)	47.05s
09	2257 (4569)	11.56s	4508 (9234)	13.35s	5370 (11217)	21.47s
10	3043 (6126)	32.33	4468 (9002)	44.01s	5954 (12291)	58.78s
11	3305 (6674)	18.38s	4629 (9514)	22.13s	7169 (14875)	31.87s
12	2575 (5162)	69.74s	4538 (9201)	99.08s	7052 (14446)	103.11s
13	1438 (2957)	20.87s	3742 (7841)	25.69s	6470 (13448)	36.35s
14	5140 (10376)	83.64s	8504 (17341)	96.19s	12905 (26541)	161.56s
15	3696 (7435)	121.86s	8416 (17183)	144.45s	14598 (30426)	207.70s
16	6301 (12751)	172.66s	9250 (18810)	255.15s	14439 (29858)	363.61s
17	7710 (15710)	551.31s	10724 (21915)	643.25s	14367 (29459)	763.69s
18	3902 (7858)	184.27s	6181 (12447)	208.02s	8313 (16894)	211.84s

considered for the experiment. As shown in Table 2.7, all hypergraphs are converted to corresponding graphs G using clique representation, and the weight of each edge in the clique equals to $1/(|e|-1)$, where $|e|$ is the number of vertices in the hyperedge; $|V|$ is the number of nodes in the hypergraph; $|V_S|$ is the number of nodes after graph reduction; $|E_H|$, $|E_G|$ and $|E_S|$ are the number of edges in hypergraphs, converted original graphs and the reduced graphs, respectively; $\frac{V}{V_S}$ and $\frac{E_G}{E_S}$ represent the node reduction ratio as well as the edge reduction ratio for spectral graph reduction; T_r is the time for graph reduction.

k-way spectral partitioning is performed on both generated graph G and the

Table 2.9
Performance of Spectral Hypergraph Partitioning on Reduced Graphs S.

ibm	Partitioning w/ Reduced Graph (S)					
	8-way		16-way		32-way	
	Cut (SD)	$T_e^S\left(\frac{T_e^G}{T_e^S}\right)$	Cut (SD)	$T_e^S\left(\frac{T_e^G}{T_e^S}\right)$	Cut (SD)	$T_e^S\left(\frac{T_e^G}{T_e^S}\right)$
01	637 (1295)	$0.08s(17X)$	914(1915)	$0.11s(13X)$	1471 (3130)	$0.18s(11X)$
02	1034 (2204)	$0.08s(56X)$	2173 (4654)	$0.17s(34X)$	3415 (7426)	$0.25s(33X)$
03	1549 (3124)	$0.18s(35X)$	2729 (5627)	$0.23s(32X)$	4483 (9520)	$0.33s(35X)$
04	2012 (4093)	$0.11s(49X)$	3256 (6840)	$0.18s(41X)$	4869 (10387)	$0.27s(39X)$
05	3144 (6417)	$1.17s(22X)$	4680 (9946)	$1.65s(33X)$	5606 (11237)	$2.30s(27X)$
06	1742 (3587)	$0.17s(71X)$	3460 (7264)	$0.35s(45X)$	4545 (9851)	$0.51s(41X)$
07	1643 (3404)	$0.14s(98X)$	3925 (8069)	$0.23s(81X)$	6373 (13495)	$0.29s(81X)$
08	3291 (6747)	$0.51s(63X)$	4916 (10126)	$0.75s(45X)$	6041 (12825)	$1.02s(46X)$
09	2164 (4379)	$0.25s(46X)$	3241 (6740)	$0.42s(32X)$	4467 (9163)	$0.71s(30X)$
10	3104 (6275)	$0.20s(162X)$	4847 (9794)	$0.31s(142X)$	6324 (13014)	$0.42s(140X)$
11	3220 (6504)	$0.40s(46X)$	4972 (10070)	$0.58s(38X)$	7396 (15671)	$0.78s(41X)$
12	2323 (4658)	$0.42s(166X)$	4622 (9395)	$0.56s(1774X)$	6890 (14140)	$1.15s(90X)$
13	1561 (3156)	$0.18s(116X)$	3849 (7912)	$0.25s(103X)$	6934 (14759)	$0.40s(91X)$
14	3726 (7475)	$0.24s(349X)$	7975 (16115)	$0.39s(247X)$	11779 (24282)	$0.52s(311X)$
15	3727 (7506)	$0.33s(369X)$	7431 (14926)	$0.47s(307X)$	12738 (26387)	$0.85s(244X)$
16	6279 (12646)	$0.39s(443X)$	10090 (20543)	$0.66s(387X)$	14894 (30695)	$0.91s(400X)$
17	7874 (15970)	$0.92s(599X)$	10141 (20716)	$1.17s(550X)$	16172 (33366)	$1.74s(439X)$
18	3252 (6529)	$0.35s(526X)$	5947 (12012)	$0.38s(547X)$	8128 (16610)	$0.55s(385X)$

spectrally-reduced graph S. Hyperedge cut and sum of external degrees (SOED) of all hyperedges that span multiple partitions are calculated to evaluate the partitioning quality. The overall performance of hypergraph partitioning with 8-, 16- and 32-way partitions are shown in Table 2.8 and Table 2.9 for using the original graph G and the reduced graph S , where "Cut" and "SD" denote the hyperedge cut and SOED metric, respectively; T_e^G and T_e^S denote the eigendecomposition time required by G and S, respectively. Comparing to the hypergraph partitioning using graph G, results show that partitioning with spectrally reduced graphs can produce comparable partitioning qualities with dramatically reduced cost.

2.4.5 Results of Scalable Data Visualization

We first demonstrate the connection between the t-SNE embedding solution and the first few unnormalized Laplacian eigenvectors of the k-NN graph formed using the original data set. To quantitatively estimate their correlations, we increase the number of Laplacian eigenvectors for representing the embedding vectors $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^n$ that store the locations of n data points in 2D space obtained by running t-SNE, and compute the correlation factors $p^x_{tsne} = \frac{||\mathbf{U}\mathbf{U}^\top \mathbf{x}||_2}{||\mathbf{x}||_2}$ and $p^y_{tsne} = \frac{||\mathbf{U}\mathbf{U}^\top \mathbf{y}||_2}{||\mathbf{y}||_2}$, where $\mathbf{U} \in \mathbb{R}^{n \times r}$ is the matrix with the first r Laplacian eigenvectors (of the original k-NN graph) as its column vectors. If p^x_{tsne} or p^y_{tsne} is close to 1, it indicates a strong correlation (significant overlap) between the eigenspace formed by the first few Laplacian eigenvectors and the t-SNE embedding vectors. Figure 2.15 shows strong correlations between the low-dimensional embedding vectors generated by t-SNE and the first few (e.g., $r = 20$) eigenvectors of the Laplacian matrices corresponding to the k-NN graphs constructed using the USPS and MNIST data sets[1]. It is also interesting to observe that the t-SNE embedding vectors are more closely related to the 10-th eigenvector, since the inclusion of such an eigenvector leads to significantly improved correlation factors p^x_{tsne} and p^y_{tsne}. This is actually very reasonable considering the ground-truth number of clusters for the USPS and MNIST data sets is 10.

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

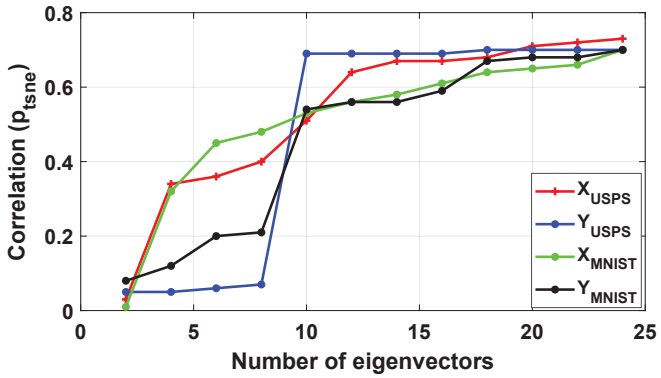


Figure 2.15: Correlations (X_{USPS} and X_{MNIST} for p^x_{tsne}; Y_{USPS} and Y_{MNIST} for p^y_{tsne}) between 2D embedding vectors computed by t-SNE and the subspace formed by the first few eigenvectors of the Laplacian matrices computed using USPS and MNIST data sets.

We also demonstrate the t-SNE visualization results obtained by leveraging spectrally-reduced NN graphs in Figures 2.16 and 2.17. Our results show very clear cluster boundaries after spectral graph reduction, which retain the ones obtained from the original data sets, indicating very high-quality embedding results as well as significantly improved runtime performance.

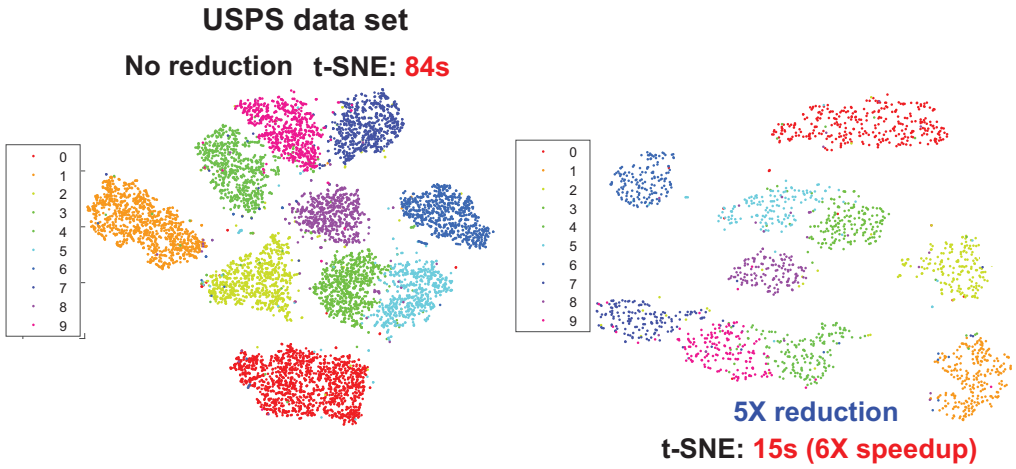


Figure 2.16: t-SNE visualization with original USPS data set and the reduced data set.

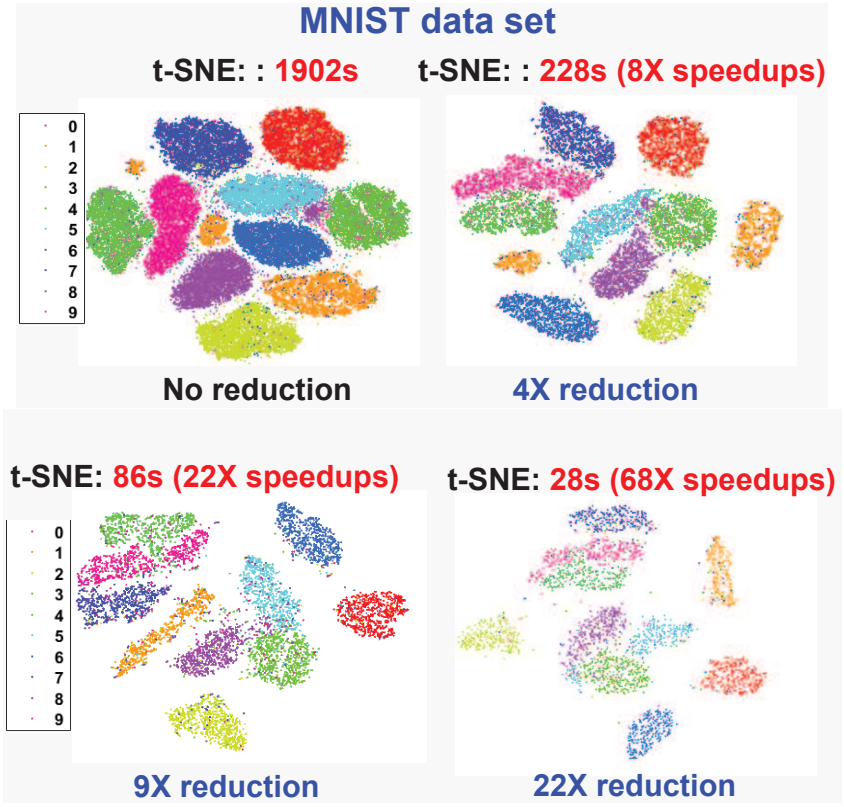


Figure 2.17: t-SNE visualization with original MNIST data set and data sets under different reduction ratios.

Chapter 3

SAMG: Sparsified Graph-Theoretic Algebraic Multigrid for Solving Large Symmetric Diagonally Dominant (SDD) Matrices

3.1 Background

Algebraic Multigrid (AMG) [82] solvers have been developed for solving large sparse matrices based on multigrid principles. Compared to geometric multigrid (GMG)

solvers that rely on the geometric information of underlying problems, AMG solvers build hierarchical coarse level problems using the graph information extracted from input matrices. A good AMG solver should be not only fast and scalable but also reliable and robust for different kinds of input matrices. The Combinatorial Multigrid solver (CMG) [52] and Lean Algebraic Multigrid solver (LAMG) [64] are state-of-the-art graph-theoretic AMG solvers that exploit spectral properties of graph Laplacian matrices to achieve high efficiency and robustness.

CMG is a highly-efficient graph-theoretic AMG solver for computer vision and image processing applications. CMG forms coarse level graphs by a graph decomposition procedure similar to the construction of a quotient graph [52]. However, the coarse level problems (graphs) obtained by CMG can be dense and may lead to dramatically increased computational cost. For example, during the CMG setup phase, we observed that for some relatively dense input SDD matrices, the graph densities of coarse level problems would grow very fast, which can significantly impact the CMG solver speed as well as efficiency for parallel computing (due to the high communication cost).

Another well known graph-theoretic AMG solver is the LAMG solver [64] whose setup phase contains two main steps: 1) low-degree node elimination and 2) node aggregation based on node proximity (algebraic distance). It also integrates a lean piecewise-constant interpolation step and an energy correction scheme to improve the

overall convergence. It is shown that the LAMG solver can achieve $O(m)$ time and storage efficiency during the setup phase and requires $O(m \log(1/\epsilon))$ operations for achieving an accuracy level ϵ during the iterative solution phase. Although LAMG is usually slower than the CMG solver for sparse matrices obtained from computer vision and image processing applications, it is usually more reliable and applicable to a broader range of applications [64]. However, the LAMG solver may run into a similar issue as the CMG solver according to our observations: high graph densities of coarse level problems may introduce rapidly increasing computational cost, which can significantly impact its efficiency.

3.2 Sparsified Algebraic Multigrid

3.2.1 Overview of Our Method

The proposed SAMG solver in this work is built upon the framework of the prior LAMG solver [64]. During the SAMG setup phase, we introduce a graph sparsification procedure based on a recent spectral perturbation based spectral graph sparsification approach [27] to effectively control the graph density while still assuring sufficient approximation quality. To more clearly illustrate the technical contribution of this work, a comparison with the LAMG solver for the setup phase is shown in Figure

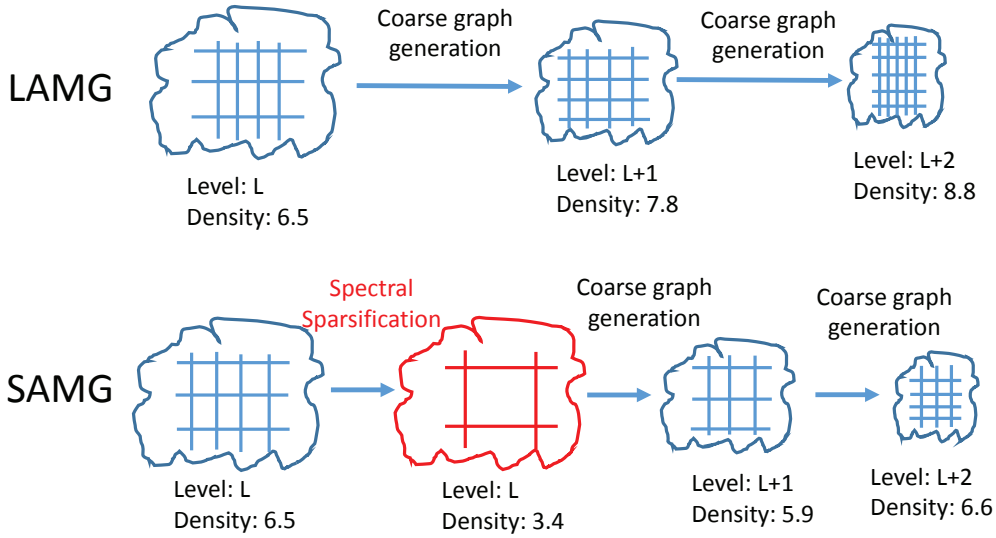


Figure 3.1: Comparison of the setup phases between LAMG [64] and SAMG (this work).

3.1. To set up a coarser level $(l+1)$ graph from an existing coarse level (l) graph in SAMG, we perform the following steps:

(1) Check the convergence rate at the current level (l) by performing a few Gauss-Seidel (GS) relaxations: if the residual drops slowly, another coarser level $(l+1)$ problem will be needed.

(2) Perform a spectral graph sparsification step if the graph density of the coarse level (l) problem is too high.

(3) Generate a coarser (level $l+1$) problem by performing node elimination and node aggregation using graph-theoretic AMG operations proposed in [64].

Although the spectral sparsification engine can preserve long-range effects in the

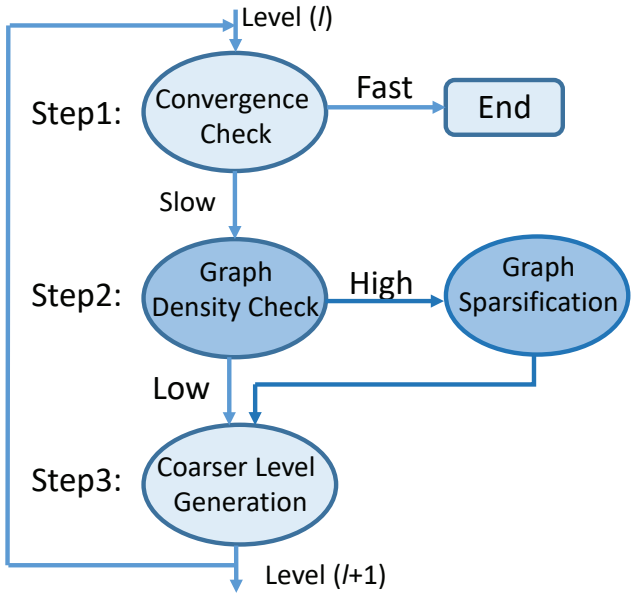


Figure 3.2: Flowchart for the SAMG solver setup phase.

graph (e.g., distance or effective resistance between nodes), it should not be used very frequently when setting up the AMG coarse level problems to assure fast converge of the AMG solver.

3.2.2 Sparsified Algebraic Multigrid (SAMG)

The complete SAMG setup flow is depicted in Figure 3.2, which includes the following key steps: (1) Convergence Check, (2.1) Graph Density Check, (2.2) Graph Sparsification, and (3) Coarser Level Generation. It should be noted that steps (1) and (3) are similar to the procedures in the prior LAMG algorithmic framework [64], while steps (2.1) and (2.2) are newly proposed in this work. We will describe above key steps in details in the following subsections.

3.2.2.1 Graph Density Check

Given a graph Laplacian matrix L_{G_l} at level l, we will first check its graph density. A graph sparsification step will be necessary if the graph density is too high. To this end, the graph size and density of each coarse level problem are considered as the key parameters for determining if a spectral graph sparsification procedure is needed according to the following observations: (a) to control coarse level graph densities for all hierarchical levels, it is more effective to sparsify finer level problems with larger sizes than sparsifying coarser level problems with smaller sizes; in other words, the spectral sparsification step should be done as early as possible for effectively reducing coarse level problem densities. (b) The spectral graph sparsification step should be performed only when the coarse level problems become increasingly denser since such a sparsification procedure will inevitably introduce approximation errors (spectral dissimilarities) that can be quantitatively measured using the relative condition number.

To efficiently identify the most suitable coarse level problem for graph sparsification, we will consider the changing rate of nonzero elements in the graph Laplacian, as well as the graph densities (the number of edges divided by the number of nodes) across different coarse level problems. Let nnz_l and agd_l denote the number of edges and the graph density for the coarse level graph G_l at level l, respectively. Then the

changing rate of edge numbers can be evaluated by:

$$\omega_l = \frac{nnz_l}{nnz_{l-1}}, \tag{3.1}$$

where ω_l reflects the changing rate of edge numbers from level $l-1$ to level l. Similarly, the changing rate of graph density can be computed by:

$$\theta_l = \frac{agd_l}{agd_{l-1}}. \tag{3.2}$$

A greater value of θ_l indicates the coarser problems are getting increasingly denser in a much faster way. Consequently, if either ω_l or θ_l is large enough, a graph sparsification procedure at the coarse level will be necessary. By defining thresholds ω_{th} and θ_{th}, the graph sparsification procedure will be performed at level l once $\omega_l > \omega_{th}$ and $\theta_l > \theta_{th}$. For the same problem, setting a larger ω_{th} or θ_{th} value will potentially allow the spectral sparsification step to be applied at a coarser level. For the proposed SAMG scheme, we observed that the optimal performance can be achieved by setting $\omega_{th} = 0.48 \sim 0.53$ and $\theta_{th} = 1.35 \sim 1.5$, respectively.

3.2.2.2 Graph Sparsification and Spectral Similarity Control

Once a coarse level problem is selected for sparsification, a graph sparsification procedure will be launched, which includes spectral similarity checking, spectral graph

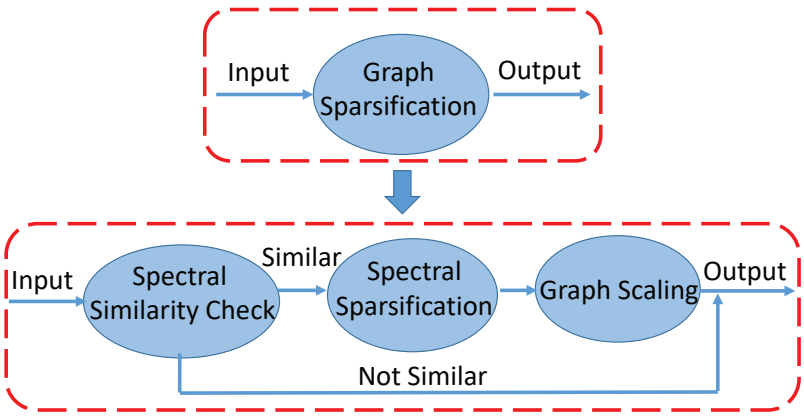


Figure 3.3: Graph sparsification during the SAMG solver setup phase.

sparsification and graph Laplacian scaling steps as illustrated in Figure 3.3.

Since each spectral graph sparsification process will introduce a "spectral gap" between the original problem and the sparsified problem, the graph sparsification procedure should not be performed very frequently to ensure fast convergence of the multigrid solver. Therefore, it is necessary to check if the existing "spectral gap" (introduced by all prior sparsification steps) still allows performing another spectral graph sparsification to the current coarse level problem during the SAMG setup phase. To this end, the relative condition numbers during the previous spectral sparsification steps will be used for estimating the "spectral gap" (Δ_l) at level l, which can be estimated by multiplying all the previous relative condition numbers. Denoting relative condition number of the sparsified graph at level s by $\kappa(L_{G_s}, L_{P_s})$, the total "spectral gap" can be computed as follows:

$$\Delta_l = \prod_{s<l} \kappa(L_{G_s}, L_{P_s}), \tag{3.3}$$

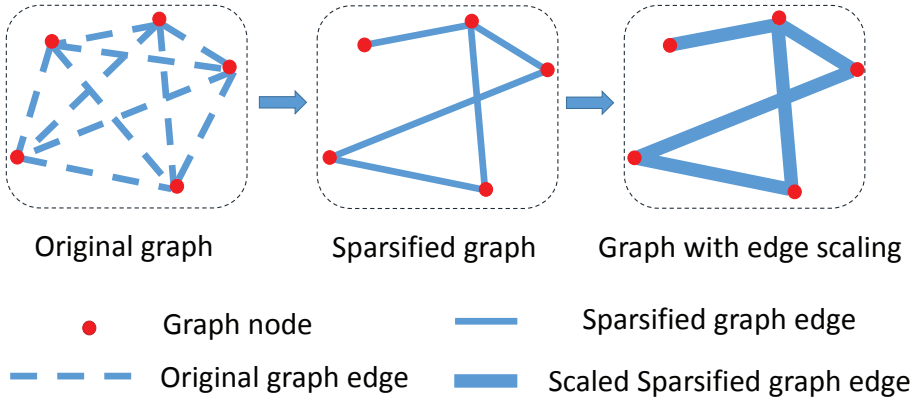


Figure 3.4: Spectral graph sparsification with graph scaling.

where L_{G_s} and L_{P_s} denote the graph Laplacian matrices of the original coarse level graph G_s and the sparsified coarse level graph P_s, respectively. Define Δ_{th} to be the threshold for the "spectral gap", then the SAMG setup phase will only allow graph sparsification steps if $\Delta_l < \Delta_{th}$; otherwise, spectral sparsification will not be applied for the following coarser levels, since a too large "spectral gap" may result in degraded convergence of the SAMG solver.

Since the sparsified graph only includes a small portion of the edges of the original coarse level graph, the total conductance of sparsified graph (sum of edge weights) is always smaller than the original graph. To compensate for the accuracy loss due to the spectral graph sparsification process, we introduce a graph scaling procedure as illustrated in the Figure 3.4 to improve the approximation quality of the sparsified graph, which scales up all the edges in the sparsified graph so that they can better

approximate the original graph. The edge scaling factor α_l for level l is computed by:

$$\alpha_l = \frac{\sum\limits_{(p,q)\in G_l} w_{p,q}}{\sum\limits_{(p,q)\in P_l} \tilde{w}_{p,q}}, \tag{3.4}$$

where $w_{p,q}$ and $\tilde{w}_{p,q}$ denote the weight of edge $(p,q) \in G_l$ and edge $(p,q) \in P_l$ in the original and sparsified coarse level graphs, respectively.

3.2.2.3 Coarser Level Generation

To generate the next coarser level problem based on the current graph, a node aggregation scheme is applied based on a node affinity metric c_{uv} that can be defined as follows for neighboring nodes u and v[64]:

$$c_{uv} = \frac{\|(X_u, X_v)\|^2}{(X_u, X_u)(X_v, X_v)}, (X_u, X_v) = \Sigma_{k=1}^{K}(x_u^{(k)} \cdot x_v^{(k)}) \tag{3.5}$$

where $X = (x^{(1)} \ldots x^{(K)})$ denotes K test vectors that are computed through applying a few GS relaxations to the linear system equation $L_{G_l}x = 0$ with different initial random vectors. The node affinity c_{uv} can effectively reflect the distance or strength of connection between nodes u and v: a larger c_{uv} value indicates a stronger connection between nodes u and v [64]. Consequently, nodes with large affinity values can be aggregated together to form the nodes on the coarser level graph.

3.3 Experimental Results

In this section, extensive experiments have been conducted to evaluate the proposed SAMG solver for different types of SDD matrices. Some of the test cases are from the SuiteSparse Matrix Collection [17], including matrices arising from IC simulations, thermal problems, finite-element analysis problems, etc. Additionally, the SDD matrices of 3D mesh grids obtained from 3D-IC thermal analysis (3D-IC_X) and image processing (Laplacian3D) are also included. We also examine the Laplacian matrices obtained from k-nearest neighbor (kNN) graphs that have been heavily studied in data mining and machine learning communities. The well known MNIST data set of handwritten digits that consist of $60,000$ images for training and $10,000$ images for testing procedures are analyzed using kNN graphs, where $k = 9, 18, 21$ are used for setting up Laplacian matrices with different graph densities.

All experiments are performed using a single CPU core of a computing platform running 64-bit RHEL 6.0 with 2.67GHz 12-core CPU and 48GB DRAM memory. The SAMG setup time for multigrid hierarchy construction is similar to the original LAMG solver [64], since the cost for spectral sparsification of coarse level problems can be negligible.

The results of the LAMG and SAMG solvers are reported in Table 3.1. The systems

Table 3.1
Experimental result of LAMG and SAMG.

Test Case	$\|V\|$	NNZ	$T_S(I_S)$	$T_L(I_L)$	(T_L/T_S)
G2_circuit	1.5E5	7.3E5	1.8s(5)	3.1s(6)	1.72X
G3_circuit	1.6E6	7.7E6	15.6s(5)	19.7s(6)	1.26X
ecology2	1.0E6	5.0E6	5.5s(4)	8.0s(4)	1.45X
thermal2	1.2E6	8.6E6	6.6s(2)	12.0s(3)	1.82X
parabolic_fem	5.3E5	3.7E6	11.8s(10)	19.8s(10)	1.68X
3D-IC_1	2.5E5	1.7E6	1.2s(4)	3.0s(5)	2.36X
3D-IC_2	5E5	3.5E6	2.3s(5)	3.9s(6)	1.69X
3D-IC_3	1E6	6.9E6	4.2s(4)	8.8s(5)	2.10X
Laplacian3D	1E6	5.0E6	4.8s(3)	7.1s(3)	1.47X
MNIST9	7.1E4	1.3E6	0.15s(1)	0.2s(1)	1.33X
MNIST18	7.1E4	2.6E6	0.18s(1)	0.46s(1)	2.55X
MNIST21	7.1E4	3.0E6	0.21s(1)	0.26s(1)	1.23X

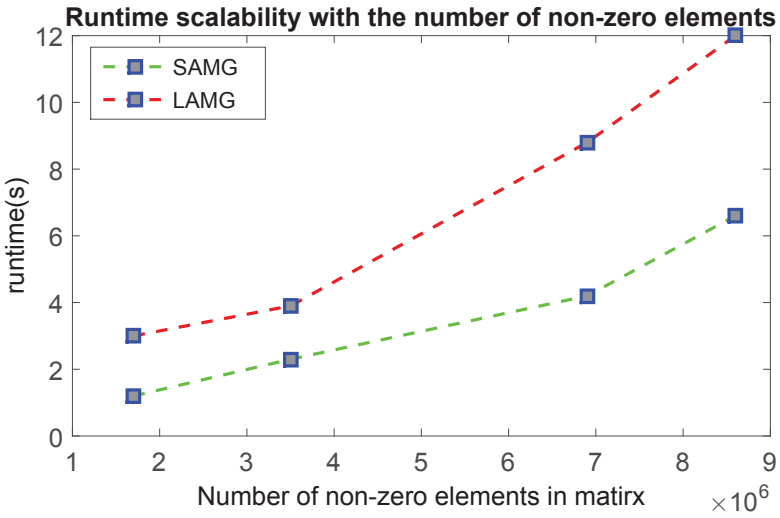


Figure 3.5: Runtime scalability with increasing number of nonzero elements.

$Ax = b$ are solved for a randomly generated right-hand-side (RHS) vector b. Both LAMG and SAMG solvers are configured to achieve the same accuracy level $||Ax - b|| < 10^{-4}||b||$. "$|V|$" represents the number of the nodes, "NNZ" denotes the number of nonzero elements in the original matrix, while "T_L" and "T_S" denote the total

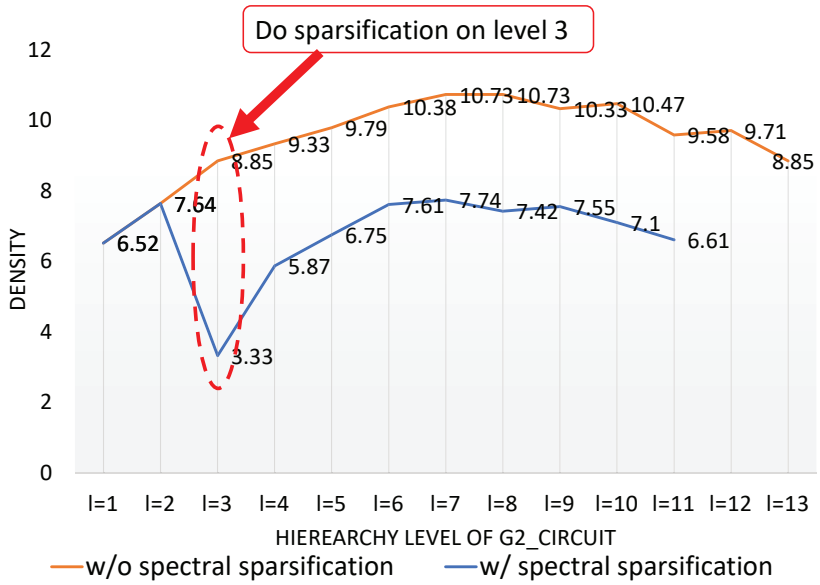


Figure 3.6: Comparison of average graph densities of coarse level problems for G2_circuit matrix.

solution time for LAMG and SAMG, respectively. I_L and I_S denote the number of multigrid iterations for LAMG and SAMG for converging to the required accuracy level, and T_L/T_S is the runtime speedup of SAMG over LAMG.

From Table 3.1, we can see that the proposed SAMG solver is substantially faster than the prior LAMG solver. The iteration numbers of SAMG and LAMG are almost the same, which indicates that the spectral sparsification steps have not influenced the convergence behavior significantly. Figure 4.12 shows the runtime scalability with respect to the nonzero elements in different matrices for LAMG and SAMG. Obviously, the runtime is almost linear with the number of nonzero elements. Figure 3.6, and Figure 3.7 show the graph densities of different coarse level problems when running the SAMG and LAMG solvers: it is observed that the graph densities of the

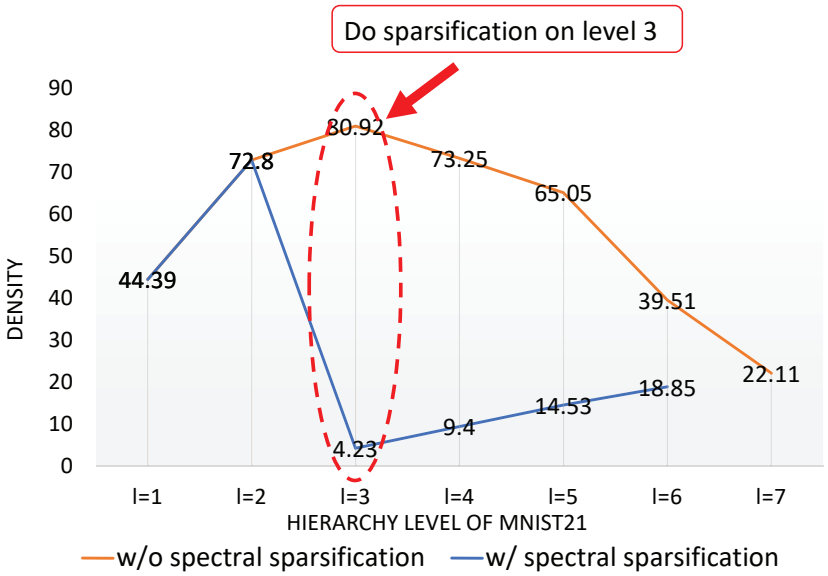


Figure 3.7: Comparison of average graph densities of coarse level problems for MNIST21.

sparsified coarse level problems in SAMG are much lower than the ones in LAMG.

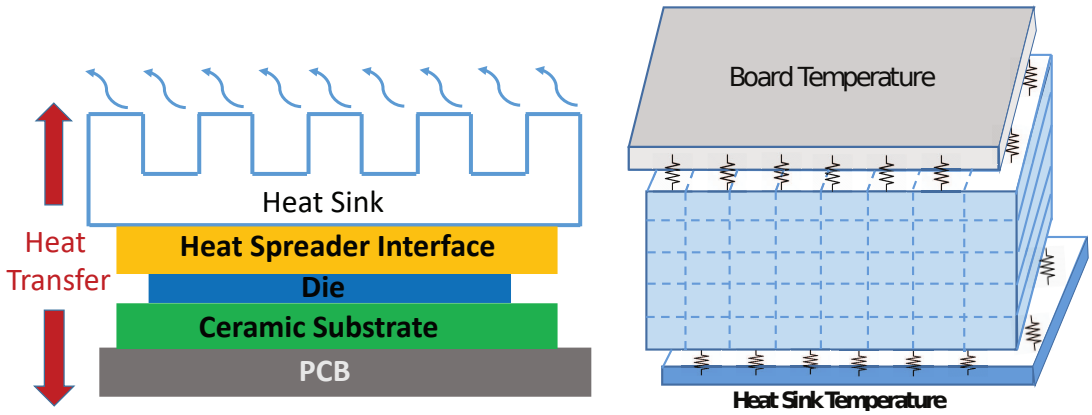
Chapter 4

A Spectral Approach to Scalable Vectorless Power Grid and Thermal Integrity Verification

4.1 Background

4.1.1 On-chip Thermal Modeling and Analysis

A diagram of an integrated chip (IC) in a C4 package is shown in Figure 4.1 (a), showing two major heat transfer paths: one is through the heat sink to the ambient



(a) Chip package with the heat sink (b) 3D modeling of the die

Figure 4.1: Thermal modeling of the chip package

surroundings, and the other is from the chip package to the board. The equivalent thermal circuit of the die is usually modeled as a 3D mesh grid with thermal conductance computed according to the materials as well as a discretization scheme, as shown in Figure 4.1 (b). The heat diffusion in an IC is modeled by the following PDE equation [60]:

$$\rho c_p \frac{\partial T(\vec{r},t)}{\partial t} = \nabla(k(\vec{r},t)\nabla T(\vec{r},t)) + p(\vec{r},t), \tag{4.1}$$

subject to the boundary condition:

$$k(\vec{r},T)\frac{\partial T(\vec{r},t)}{\partial n_j} + h_j T(\vec{r},t) = f_j(\vec{r},t), \tag{4.2}$$

where ρ is the material density (kg/m^3), c_p is the specific heat $[J/(kg \cdot °\mathrm{C})]$, T is the temperature (°C), $\vec{r}$ is the location in the 3D space, k is the thermal conductivity of the material $[W/m^2 \cdot °\mathrm{C}]$, $p(\vec{r},t)$ is the power density of heat sources (W/m^3), n_j is the outward direction normal to the boundary surface j, h_j is the heat transfer

coefficient $[W/(m^2 \cdot °\mathrm{C})]$, and f_j is an arbitrary function at the surface j.

An emerging trend is increased functionality on smaller chip areas. The increased power density will lead to an increased temperature gradient, significantly impacting on-chip performance. For example, high operating temperatures will usually lead to the increased leakage power, degraded transistor performance and increased interconnect resistivity. To ensure adequate functionality of the chip including chip timing, signal integrity and power leakage, thermal analysis is necessary and increasingly critical for designing modern integrated circuits (ICs).

4.1.2 Vectorless Power Grid and Thermal Integrity Verification

The steady-state analysis of an n-node thermal grid (or power grid) can be formulated into a system of linear equations using nodal analysis [25, 33]:

$$T\,x \;=\; b. \tag{4.3}$$

For a power grid, T is a conductance matrix representing all the interconnected resistors in the grid, x is $n \times 1$ node voltage vector, and b is the right-hand side current vector. For thermal analysis and verification, T is the thermal conductance

matrix of the 3D thermal grid, b is the right-hand-side (RHS) vector modeling the underlying workload (power density) distribution, and x is the unknown temperature vector to be computed.

Traditional vectorless power grid voltage or current integrity verification aims to identify the maximum voltage drops or current densities under linear current constraints [26, 33], where current constraints are introduced to capture current loading variations and correlations in a given chip design. Thermal integrity verification seeks to identify the maximum temperature or temperature gradients across the chip given uncertain workloads or power source configurations, similar to prior vectorless power grid integrity verification problems. There are both local and global constraints in a typical vectorless verification problem: local constraints for setting the lower and upper bounds of the power density for each source and global constraints for setting the lower and upper bounds for blocks of sources.

The proposed vectorless integrity verification tasks compute the maximum voltage drop given any power grid designs or worse-case temperatures across the chip given any thermal grids by solving the following linear program (LP) for each individual

node i:

$$\begin{aligned}
&\text{maximize}: \quad x_i \;=\; e_i^{\top}\, T^{-1}\, b, \quad \text{for} \;\; i = 1, \ldots, n \\
&\text{subject to the following constraints on power densities:} \\
&\text{Local Constraints}: \quad b^L \;\leq\; b \;\leq\; b^U, \\
&\text{Global Constraints}: \quad B^L \;\leq\; M\, b \;\leq\; B^U,
\end{aligned} \tag{4.4}$$

where n is the number of nodes in the power gird or 3D thermal grid, e_i is an elementary unit vector with $i-th$ entry equal one and the remaining entries equalling zeros. Since the conductance matrix T is an M-matrix, the T^{-1} only contains non-negative sensitivity values. The b^L (B^L) and b^U (B^U) represent the lower bounds and upper bounds of individual power sources (blocks), while M is an $m \times n$ matrix that only contains 0s and 1s for defining m global (block) constraints. After finding the worst-case vector b_{wst} through the above optimization procedure, we can simply compute the maximum voltage drop or worst-case temperature distribution x_{wst} using $x_{wst} = T^{-1} b_{wst}$.

4.1.3 Vectorless Thermal Verification Challenges

The adjoint temperature sensitivity with respect to each power source will be needed for setting up the LP problems in (4.4) for vectorless thermal verification. For example, the adjoint sensitivity vector s for computing node temperature t_i considering all power sources in b can be calculated by solving the linear system of equations

$T\,s\;=\;e_i$. Once the matrix factorization for T is computed, adjoint thermal sensitivity vectors for individual node temperatures can be efficiently obtained by reusing the matrix factors.

However, **factorization of the thermal matrix** obtained from 3D mesh-structured grids can be much more costly than factorizing the conductance matrices for power grid vectorless verification tasks [107, 113], due to the much faster-growing computational complexity of existing direct solution methods, such as LU and Cholesky decomposition methods [16]. For example, our results show that factorizing a matrix with one million rows (columns) using the state-of-the-art Cholesky solver [16] may take over 30 minutes and consume 18GB memory.

Further, since the adjoint sensitivity vector is needed for solving the following LP problem:

$$\text{maximize}: \quad t_i \;=\; \sum s_i\, b_i, \tag{4.5}$$

very high computational complexity will be expected when a large number of uncertain power sources (variables) are involved.

4.1.4 Graph Signal Processing and Spectral Sparsification

There is an analogy between traditional signal processing (classical Fourier analysis) and graph signal processing [87]: **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. The recent spectral graph sparsification process [27, 28] aims to maintain as few as possible edges for preserving the slowly-varying or "low-frequency" signals of the original graphs, which therefore can be regarded as a "low-pass" graph filter. As a result, spectrally-sparsified graphs will be able to preserve the eigenvectors associated with low eigenvalues more accurately than high eigenvalues.

To aggressively simplify the 3D thermal grids and thereby addressing the computational challenges in vectorless integrity verification without sacrificing the approximation accuracy, emerging graph signal processing and spectral graph sparsification research can be leveraged [28, 87]. Since full-chip temperature distributions can be considered as the "low-frequency" graph signals on thermal grids obtained after applying a "low-pass" graph filter on the original input power sources, the spectrally-sparsified thermal grids will well-preserve the temperature distributions.

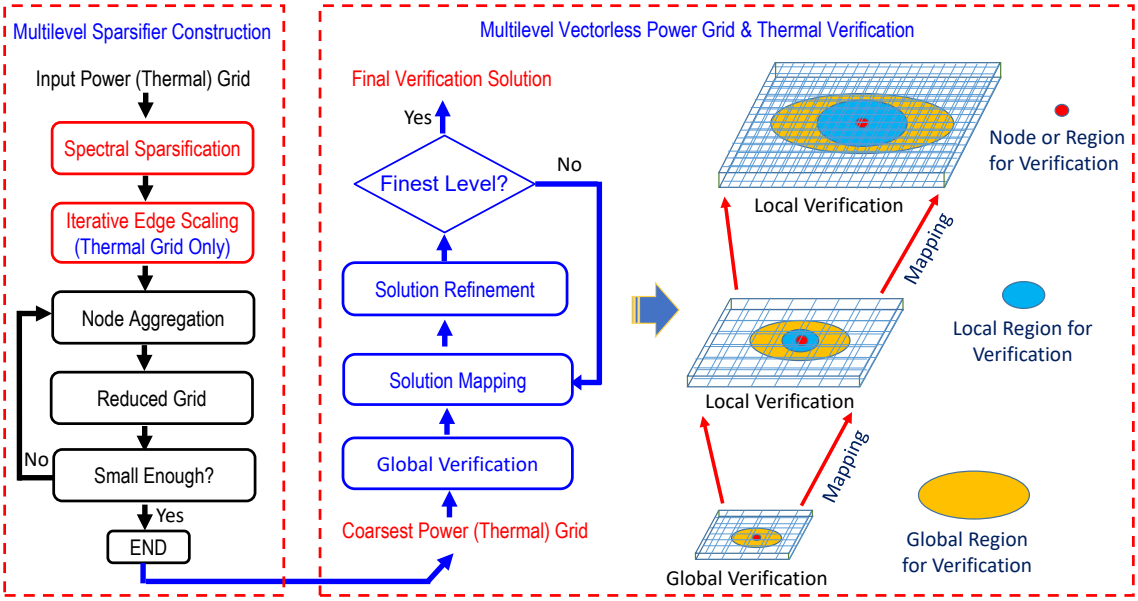


Figure 4.2: Multilevel vectorless power grid and thermal integrity verification.

4.2 A Spectral Approach to Vectorless Power Grid and Thermal Integrity Verification

4.2.1 Multilevel Verification Framework

In this work, we propose a multilevel vectorless verification framework shown in Figure 4.2. Our approach is based on the latest graph-theoretic algebraic multigrid (AMG) research [64, 116] for generating coarse-level (sparsified) grids according to the original power (thermal) grid problem, as well as the recent multilevel vectorless power grid

integrity verification framework [25, 113]. The proposed framework includes two phases: (a) a setup phase for creating multilevel power (thermal) grids and their sparsified grids and (b) a multilevel vectorless verification phase for identifying worst-case voltage drop or thermal scenarios.

Phase (a) includes the following key steps:

1. Perform spectral sparsification for the original power (thermal) grid to reduce network complexity without changing the grid size.

2. For thermal grids, do iterative edge weight scaling scheme to update the sparsified grid.

3. Apply nodal aggregation on the sparsified grid to recursively generate hierarchical grid sparsifiers.

4. Define AMG-like restriction (prolongation) operators for constraints (solution) mapping operations at different level based on the nodal elimination and aggregation schemes obtained at the previous step,

5. Factorize the grid sparsifiers at each level for adjoint sensitivity computations.

Phase (b) includes the following key steps for vectorless verification of a specific node or grid region:

1. Compute adjoint sensitivities using the matrix factors at each level.

2. Identify a critical global region on the coarsest level and local critical regions on finer levels.

3. Set up LPs at the coarsest level to obtain the initial solution vector.

4. Recursively map solution vectors to the next finer level and improving the solution accuracy by performing local solution refinements at each level until reaching the finest level.

4.2.2 Spectral Sparsification and Scaling of 3D Thermal Grids

To substantially reduce the cost for the matrix factorization and LP solution phases, we exploit a perturbation-based spectral graph sparsification engine [27, 28] to sparsify the topology of the original 3D thermal grid. The spectral sparsification step can effectively control the thermal grid densities while maintaining good spectral approximation that is critical for accurate vectorless verification tasks: the sparsified thermal grids have tree-like structures that will immediately reduce the matrix factorization time while preserving the effective thermal resistances between nodes. It is noted that preserving effective resistance is equivalent to preserving the adjoint sensitivities to be applied for setting up LPs. Therefore, the adjoint sensitivity for

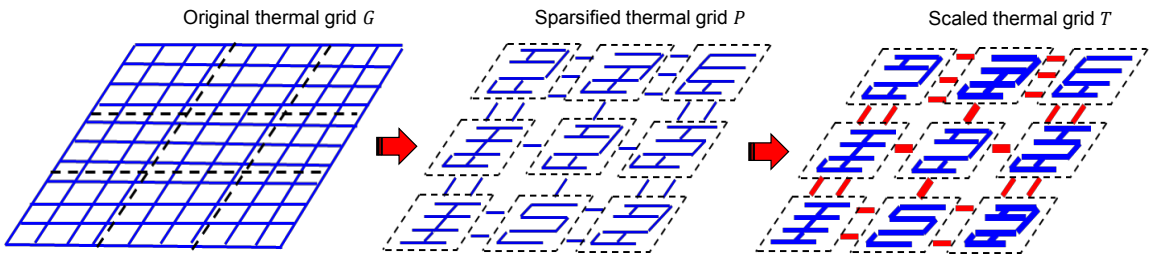


Figure 4.3: Iterative edge scaling for sparsified thermal grids.

each LP task can be computed in a more efficient way without sacrificing the final solution quality (e.g., worst-case vector). Additionally, the sparsified thermal grids will have many low-degree nodes that can be potentially merged together to further reduce the number of variables in LPs, reducing the cost for solving LPs in vectorless verification tasks.

To further improve the approximation quality of the sparsified thermal grid, we introduce an iterative edge weight scaling scheme to gradually scale up the edge weight in the sparsified thermal grid, which has been shown in Figure 4.3. This scheme will compensate for the thermal conductance loss due to the missing edges by matching the "low-frequency" behaviors of the original thermal grids, which is motivated by recent graph signal processing techniques [87].

Denote $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ the n eigenvalues of the Laplacian matrix L_G for a connected graph G with the corresponding eigenvectors denoted by $u_1, u_2, \cdots, u_n$. The spectral decomposition of the Laplacian matrix of graph G can be expressed as

Algorithm 7 Algorithm for Iterative Edge Weight Scaling

Input: The error tolerance ϵ, the number of partitions k, the original graph $\mathbf{G}$ and the initial spectrally-sparsified graph $\mathbf{P^{(0)}}$.

Output: The spectrally-sparsified graph $\mathbf{P}$ with scaled edge weights.

1: Generate a random vector $\mathbf{b}$ that is orthogonal to the all-one vector.
2: Partition the original graph $\mathbf{G}$ into $\mathbf{k}$ blocks $\mathbf{P_1}, \mathbf{P_2}, \cdots, \mathbf{P_k}$ using multilevel graph partitioning method [54].
3: Construct matrices $\mathbf{T_G} = \mathbf{L_G} + \mathbf{g_{min}}\mathbf{I}$ and $\mathbf{T_{P^{(0)}}} = \mathbf{L_{P^{(0)}}} + \mathbf{g_{min}}\mathbf{I}$ by adding a small value $\mathbf{g_{min}}$ similar to the ambient thermal conductance to each diagonal entry of $\mathbf{L_G}$ and $\mathbf{L_{P^{(0)}}}$ for graph signal filtering purpose.
4: Solve $\mathbf{T_G}\,\mathbf{x} = \mathbf{b}$ and $\mathbf{T_{P^{(0)}}}\,\widetilde{\mathbf{x}} = \mathbf{b}$ and compute $\mathbf{err} = \frac{\|\mathbf{x}-\widetilde{\mathbf{x}}\|}{\|\mathbf{x}\|}$.
5: **while** $\mathbf{err} > \epsilon$ **do**
6: **for** partition $\mathbf{P_i}$, $\mathbf{i} = \mathbf{1}, ..., \mathbf{k}$, **do**
7: calculate $\mathbf{y_i} = \sum_{\mathbf{t}\in\mathbf{P_i}} \mathbf{x}[\mathbf{t}]$, $\widetilde{\mathbf{y}}_\mathbf{i} = \sum_{\mathbf{t}\in\mathbf{P_i}} \widetilde{\mathbf{x}}[\mathbf{t}]$, and $\alpha_\mathbf{i} = \frac{\widetilde{\mathbf{y}}_\mathbf{i}}{\mathbf{y_i}}$ for all nodes;
8: **end for**
9: **for** all edges $(\mathbf{p}, \mathbf{q}) \in E_s$ **do**
10: **if** $\mathbf{p}, \mathbf{q} \in P_i$, scale up $\mathbf{w_{p,q}}$ by a factor of $\alpha_\mathbf{i}$;
11: **if** $\mathbf{p} \in P_i$ and $\mathbf{q} \in P_j$, scale up $\mathbf{w_{p,q}}$ by a factor of $(\alpha_\mathbf{i} + \alpha_\mathbf{j})/\mathbf{2}$;
12: **end for**
13: Update $\tilde{\mathbf{P}}$, $\mathbf{L_{\tilde{P}}}$ and $\mathbf{T_{\tilde{P}}}$ with the latest edge weights;
14: Solve $\mathbf{T_{\tilde{P}}}\,\widetilde{\mathbf{x}} = \mathbf{b}$ and update the mismatch $\mathbf{err} = \frac{\|\mathbf{x}-\widetilde{\mathbf{x}}\|}{\|\mathbf{x}\|}$;
15: **end while**
16: Return the latest spectrally-sparsified graph $\mathbf{P}$.

follows:

$$L_G = \sum_{i=1}^{n} \lambda_i \, u_i \, u_i^\top . \tag{4.6}$$

Adding a small grounded thermal conductance g_{min} to each node in graph G or equivalently a small element g_{min} to each diagonal element in L_G leads to:

$$T_G = L_G \, + \, g_{min} I = \sum_{i=1}^{n} (g_{min} + \lambda_i) \, u_i \, u_i^\top , \tag{4.7}$$

where the identify matrix $I = \sum_{i=1}^{n} u_i \, u_i^\top$. When expressing a random vector b using

Laplacian eigenvectors, we have:

$$b = \sum_{i=1}^{n} \beta_i u_i^{\top}. \tag{4.8}$$

Solving $T_G x = b$ is equivalent to computing $x = T_G^{-1} b$, which can be further expressed as:

$$\begin{aligned} x = (L_G + g_{min} I)^{-1} b &= \left(\sum_{i=1}^{n} (g_{min} + \lambda_i)\, u_i\, u_i^{\top} \right)^{-1} b \\ &= \sum_{i=1}^{n} \frac{u_i\, u_i^{\top} b}{g_{min} + \lambda_i} = \frac{1}{g_{min}} \sum_{i=1}^{n} \frac{\beta_i u_i}{1 + r\, \lambda_i} \end{aligned} \tag{4.9}$$

where $r = 1/g_{min}$. (4.9) indicates that when using a small g_{min}, the eigenvectors associated with small eigenvalues or only "low-frequency" components in b will remain in x; on the other hand, a relatively large g_{min} ($r \approx 0$) will allow more higher frequencies to be included in x and thus lead to $x \approx b$.

Based on the above analysis, we can consider T_G^{-1} as a "low-pass" filter that filters graph signals b. By properly choosing g_{min}, it is possible to filter out the graph signal's "high-frequency" (highly-oscillating) components and only keep "low-frequency" components in x. Since chip temperature distributions mainly contain "slowly-varying" "low-frequency") components due to relative small ambient thermal conductance values, it is thus possible to exploit emerging spectral sparsification techniques [27, 28, 29] to retain a small number of edges in the sparsified thermal grids while still preserving

accurate thermal profiles, since spectrally-sparsified graphs can well-preserve "low-frequency" graph signals. Based on the above intuition, Algorithm 7 is proposed for scaling up edge weights in the sparsified thermal grid by matching the "low-frequency" responses filtered by the original thermal grids.

4.2.3 Spectral Solution Refinement.

We define $0 = \tilde{\lambda}_1 \leq \tilde{\lambda}_2 \leq \cdots \leq \tilde{\lambda}_n$ to be the eigenvalues of L_P for sparsified graph P with the corresponding eigenvectors $\tilde{u}_1, \tilde{u}_2, \cdots, \tilde{u}_n$, such that the spectral decomposition of L_P with eigenvalues and eigenvectors can be expressed as

$$L_P = \sum_{i=1}^{n} \tilde{\lambda}_i \, \tilde{u}_i \, \tilde{u}_1^{\top}. \tag{4.10}$$

Assume that the k smallest eigenvalues and corresponding eigenvectors of L_G can be well preserved in L_P, while the remaining higher eigenvalues and eigenvectors are not. Then the spectral decomposition of L_P can be approximately written as:

$$L_P \approx \sum_{i=1}^{k} \lambda_i \, u_i \, u_i^{\top} + \sum_{k+1}^{n} \tilde{\lambda}_i \, \tilde{u}_i \, \tilde{u}_i^{\top}. \tag{4.11}$$

Based on (4.9), the solution from the original grid can be expressed as:

$$x = (L_G + g_{min} I)^{-1} b = \sum_{i=1}^{n} \frac{u_i \, u_i^\top b}{g_{min} + \lambda_i} \tag{4.12}$$

where identify matrix I can be written as follows:

$$I = \sum_{i=1}^{n} u_i \, u_i^\top \approx \sum_{i=1}^{k} u_i \, u_i^\top + \sum_{i=k+1}^{n} \tilde{u}_i \, \tilde{u}_i^\top \tag{4.13}$$

Similarly, the solution $\tilde{x}$ obtained with L_P can be written as:

$$\begin{aligned} \tilde{x} = (L_P + g_{min} I)^{-1} b &= \sum_{i=1}^{n} \frac{\tilde{u}_i \, \tilde{u}_i^\top b}{g_{min} + \tilde{\lambda}_i} \\ &\approx \sum_{i=1}^{k} \frac{u_i \, u_i^\top b}{g_{min} + \lambda_i} + \sum_{i=k+1}^{n} \frac{\tilde{u}_i \, \tilde{u}_i^\top b}{g_{min} + \tilde{\lambda}_i} \end{aligned} \tag{4.14}$$

Based on (4.12) and (4.14), the solution error due to spectral sparsification and scaling becomes:

$$\Delta x = x - \tilde{x} \approx \sum_{i=k+1}^{n} \left(\frac{u_i \, u_i^\top b}{g_{min} + \lambda_i} - \frac{\tilde{u}_i \, \tilde{u}_i^\top b}{g_{min} + \tilde{\lambda}_i} \right), \tag{4.15}$$

indicating that the solution error using spectrally-sparsified graphs can be expressed as a linear combination of eigenvectors corresponding to large Laplacian eigenvalues. In other words, the error is a linear combination of high frequency signals on graphs, which can be efficiently filtered out by using "low-pass" graph signal filters [87]. To further improve the solution obtained on sparsified thermal grids, weighted Jacobi

iterative method is used, which has been described in Algorithm 8. The inputs to our algorithm include the original thermal conductance matrix T_o that can be decomposed into a diagonal matrix D_o and the remainder matrix R_o, the solution vectors $\tilde{x}_1$,..., $\tilde{x}_k$ obtained by solving (4.14) using the sparsified thermal conductance matrix T_s, the RHS vectors b_1,..., b_k as well as the weight factor γ and iteration number N_{max} for signal filtering.

Algorithm 8 Solution Refinement Algorithm

Input: $\mathbf{T}_o = \mathbf{D}_o + \mathbf{R}_o$, $\tilde{x}_1$,..., $\tilde{x}_k$, b_1,..., b_k, γ, N_{max};

1: **for** $j = 1$ **to** k **do**
2: **for** $i = 1$ **to** N_{max} **do**
3: $\tilde{x}_j^{(i+1)} = (1-\gamma)\tilde{x}_j^{(i)} + \gamma \mathbf{D}_o^{-1}(b_j - \mathbf{R}_o \tilde{x}_j^{(i)})$
4: **end for**
5: **end for**
6: Return the smoothed solution vectors $\tilde{x}_1$,..., $\tilde{x}_k$.

4.2.4 Example: A Two-level Verification Framework

In these examples, we describe a two-level vectorless verification approach; multilevel schemes can be constructed in a similar way.

Two-level local and global constraints mapping. Local power constraints can be directly mapped from fine level h to coarse level H using AMG's restriction operator

R_h^H obtained as follows:

$$\text{Upper bound}: \quad b_H^U \;=\; R_h^H \, b_h^U \;,$$

$$\text{Lower bound}: \quad b_H^L \;=\; R_h^H \, b_h^L,$$

where b_H^U, b_H^L, b_h^U and b_h^L denote the upper bound and lower bound of power sources for coarse and fine grids, respectively. The global constraints mapping can be defined in a similar manner by choosing the global constraints on the coarse grid to be the sum of each block's lower and upper bounds on the fine grid.

Two-level Solution mapping and refinement. To reduce the verification cost on the coarse level, the global critical region C_{glb}, a set of nodes, will be identified based on the adjoint sensitivity threshold [25]. Specifically, given a sensitivity threshold ϵ_{th}, we will only include the power sources that have adjoint sensitivity values greater than ϵ_{th} into C_{glb} when setting up the LPs:

$$\text{maximize}: \quad t_{wst} \;=\; \sum_{\forall\, b_i \,\in\, C_{glb}} s_i \, b_i \tag{4.16}$$

subject to local and global constraints:

$$b^L \;\leq\; b \;\leq\; b^U, \quad B^L \;\leq\; M\, b \;\leq\; B^U. \tag{4.17}$$

The solution b^H_{wst} will be further mapped back to the fine level using the AMG prolongation operator R^h_H by $\widetilde{b}^h_{wst} = R^h_H\, b^H_{wst}$. To control the error introduced during the mapping process, a local solution refinement procedure at the fine level will be applied to incrementally improve the solution quality on the fine grid. Specifically, we set up a new LP for a much smaller local critical region.

Algorithm flow and complexity. The multilevel vectorless integrity verification algorithm is described in Algorithm 9. The key steps for each level grid consists of:

(1) Scale up the sensitivity threshold $\epsilon_{loc} = \beta\, \epsilon_{glb}$ with the scaling factor $\beta > 1$ to obtain a much smaller local critical region C_{loc}.

(2) Set up a new LP problem for the local critical region C_{loc} to obtain the solution vector $\overline{b}^h_{wst}$.

(3) Update solution for C_{loc} with $\overline{b}^h_{wst}$; Reuse the interpolated solution $\widetilde{b}^h_{wst}$ for the sources that belong to C_{glb} but not C_{loc}.

The complexity for setting up multilevel problems is $O(m)$, where m denotes the number of resistors in the chip model. The complexity for input grid spectral sparsification and edge scaling is $O(m \log n)$ with n denoting the number of nodes in the grid, which is nearly linear. The cost for solving LPs will depend on the algorithm to be adopted as well as the sizes of critical regions for setting up the LPs, which can be well controlled by taking advantage of the proposed multilevel verification framework. It should be noted that by leveraging the proposed solution refinement procedure, only

ultra-sparse (tree-like) spectral sparsifiers of the original grids are needed for vectorless verification, which can significantly improve the overall algorithm scalability, as shown in our experiment results in Section 4.3.

Algorithm 9 Multilevel Vectorless Integrity Verification

Input: original thermal or power grid, user-defined local and global power constraints $\mathbf{b^U}$, $\mathbf{b^L}$ and $\mathbf{M}$, initial normalized sensitivity threshold $\epsilon_{\mathbf{th}}$, and sensitivity scaling factor $\beta > \mathbf{1}$
Output: worst-case voltage drop of the power grid or thermal profile of the original thermal grid.

1: Extract spectrally sparsified grid for the original input grid.
2: For thermal grids, update sparsified grid using iterative edge weight scaling method (Algorithm 7).
3: Multilevel coarse grid construction:
(a) Construct all hierarchy levels from finest to coarsest level;
(b) Get local and global power constraints $\mathbf{b^U}$, $\mathbf{b^L}$ and $\mathbf{M}$ for each level using AMG mapping operators.
4: Factorize each coarse-level grid for adjoint sensitivity calculation.
5: Perform global verification at the coarsest level $\mathbf{K}$:
(a) Find global critical region $\mathbf{C^K_{glb}}$ for a given sensitivity threshold $\epsilon_{\mathbf{K}}$, and set up LP to get worst case vector $\mathbf{b^K_{wst}}$
6: Perform solution mapping and refinement on finer to finest levels:
7: $\mathbf{k \leftarrow K}$
8: **while** $\mathbf{k > 1}$ **do**
9: Interpolate solution vector to finer level by: $\tilde{\mathbf{b}}^{\mathbf{k-1}}_{\mathbf{wst}} = \mathbf{R}^{\mathbf{k-1}}_{\mathbf{k}}\mathbf{b}^{\mathbf{k}}_{\mathbf{wst}}$
10: Set sensitivity threshold $\epsilon_{\mathbf{k-1}} = \beta\epsilon_{\mathbf{k}}$ and identify $\mathbf{C}^{\mathbf{k-1}}_{\mathbf{loc}}$.
11: Setup a new LP for $\mathbf{C}^{\mathbf{k-1}}_{\mathbf{loc}}$ to obtain solution vector $\bar{\mathbf{b}}^{\mathbf{k-1}}_{\mathbf{wst}}$.
12: Combine the latest LP and interpolated solutions to form $\mathbf{b}^{\mathbf{k-1}}_{\mathbf{wst}}$.
13: $\mathbf{k \leftarrow k - 1}$
14: **end while**
15: Calculate the worst-case voltage drop or hermal distribution using the worst-case power source vector.

4.3 Experimental Results

In this section, we present the experiment results of the proposed vectorless power grid verification method on different power grid designs and thermal verification method for two microprocessor designs [60]. The proposed multilevel vectorless integrity verification method has been implemented in MATLAB and C++. The LP problems are solved by the state-of-the-art LP solver [74], and all experiment results have been obtained using a single CPU core of a computing platform running 64-bit RHEL 6.0 with 2.67GHz 12-core CPU and 48GB DRAM memory.

4.3.1 Experimental Setup

The test cases used for power grid verification include industrial power gird designs with different sizes up to 9 million nodes [71, 108]. The design details of the two microprocessors used for generating the thermal grids are shown in Table 4.1. The heat conductance paths are modeled using equivalent heat transfer coefficients.

The specifications of the power grid and thermal test cases are shown in Table 4.2, where $N.\#$ and $P.\#$ denote the numbers of grid nodes and power sources, respectively. $L.\#$ represents the number of levels generated when using the multilevel verification

Table 4.1
Statistics of two microprocessor designs

Design	Processor A	Processor B
Power Consumption (W)	28	50
Die Area (mm^2)	195	302
Num. of Metal Layers	4	6
Num. of Material Layers	11	15
Equivalent Heat Transfer	3.3 (heat sink)	3.3 (heat sink)
Coefficients ($10^3 W/m^2K$)	2.0 (package)	2.0 (package)

Table 4.2
Specifications of the power grid test cases and thermal test cases.

Power Grid Specs.				Thermal Grid Specs.			
CKT	$N.\#$	$P.\#$	$L.\#$	CKT	$N.\#$	$P.\#$	$L.\#$
*ibmpg*3	$0.85M$	$90K$	2	T1	$25K$	$2.5K$	2
*ibmpg*4	$1.0M$	$100K$	2	T2	$0.1M$	$10K$	2
*ibmpg*6	$1.7M$	$170K$	2	T3	$0.2M$	$10K$	2
*ibmpg*7	$1.5M$	$150K$	2	T4	$0.4M$	$40K$	2
*thupg*1	$5.0M$	$500K$	3	T5	$0.9M$	$90K$	2
*thupg*2	$9.0M$	$900K$	3	T6	$1.6M$	$0.16M$	2
-	-	-	-	T7	$2.0M$	$0.20M$	2

methods. When setting up the experiments, three methods for vectorless power grid and thermal integrity verification are applied, including a single level (direct) method, multilevel method without sparsification, and the proposed multilevel method with sparsification.

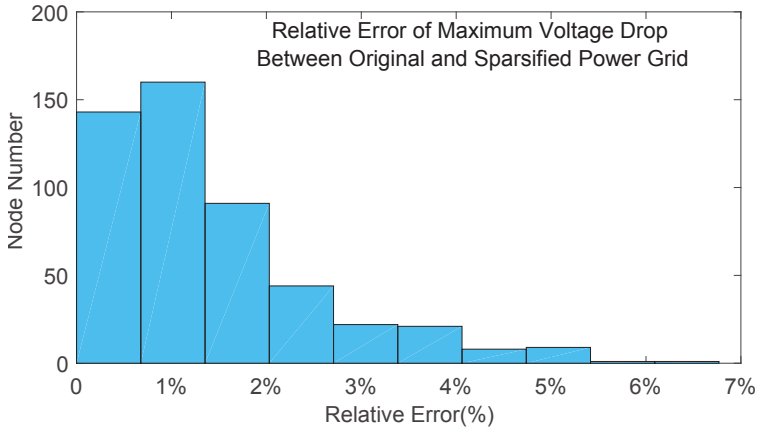


Figure 4.4: Relative error of vectorless verification w/ sparsified grid.

4.3.2 Experimental Results for Power Grid Verification

4.3.2.1 Result of Solution Quality

As we mentioned in the previous sections, the spectral graph sparsification method can well-preserve the effective resistances of the original power grid, which will guarantee a good solution quality during vectorless verifications. Figure 4.4 shows that the relative errors for the vectorless verifications with the sparsified power grids (single level) are realtively low.

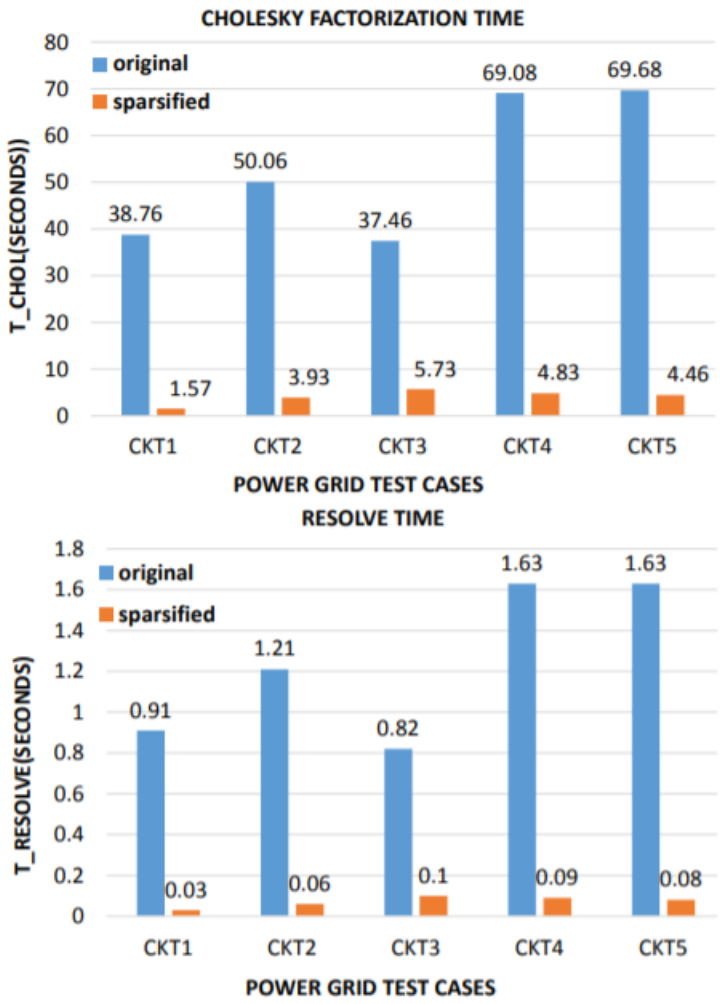


Figure 4.5: Sensitivity computation time for the original and sparsified grids.

4.3.2.2 Result of Runtime Efficiency

Adjoint sensitivity calculation for vectorless power grid verification requires matrix factorization and a linear solver using the matrix factors. For example, consider a power grid conductance matrix G. To find the voltage sensitivity at a specific node

Table 4.3
Results of the proposed vectorless power grid integrity verification method.

CKT	Single Level			Multilevel w/o Sparsification				Multilevel w/ Sparsification				
	T^o_{chol}	T^o_{sol}	T^o_{lp}	T^m_{chol}	T^m_{sol}	T^m_{lp}	$Err(\%)$	T^s_{chol}	T^s_{sol}	T^s_{lp}	$Err(\%)$	κ
ibmpg3	17.4s	1.39s	1.79s	37.12s	0.87s	0.21s	2.34%	1.40s	0.029s	0.11s	2.58%	27
ibmpg4	22.4s	1.57s	2.10s	48.04s	1.16s	0.35s	3.42%	3.72s	0.06s	0.08s	2.26%	39
ibmpg6	16.3s	2.84s	3.19s	30.30s	0.67s	0.45s	1.23%	5.41s	0.10s	0.20s	2.27%	24
ibmpg7	30.3s	2.44s	3.15s	66.50s	1.57s	0.32s	3.98%	4.86s	0.08s	0.15s	1.00%	36
thupg1	92.2s	9.20s	11.43s	433.54s	5.06s	27.15s	1.00%	11.09s	0.21s	10.03s	1.00%	42
thupg2	963.2s	43.16s	282.30s	2527.46s	398.05s	45.10s	1.00%	47.23s	0.46s	17.93s	1.00%	41

i with respect to all current sources, the right-hand-side (RHS) vector is set to be $b = [0, \ldots, 1 \ldots, 0, \ldots, 0]$ with only the i-th entry being 1 and other entries being 0. By solving the linear system of equations based on matrix factors, the solution x can be obtained and used as the sensitivity vector for setting up the following LPs. The matrix factors only need to be computed once for each level and will be reused many times during the vectorless verification process. Since the conductance matrix of a sparsified power grid can be factorized and solved in almost linear time, adjoint sensitivity computations using sparsified grids can be much more efficient than the original grid problem. As shown in Figure 4.5, the runtime results of sensitivity calculation for the original and sparsified power grids are summarised, where t_{chol} denotes the runtime for Cholesky matrix factorization, $t_{resolve}$ denotes the runtime for linear equation solving. It shows dramatic speedups in sensitivity calculations with the proposed spectral graph sparsification procedure.

Comprehensive verification results using different approaches are shown in Table 4.3, with speedup numbers shown in Table 4.4. "Single Level", "Multilevel w/o Sparsification", and "Multilevel w/ Sparsification" denote the verification methods using

Table 4.4
Runtime results of the proposed method.

CKT	Sp. Over Single Level			Sp. Over Original Multilevel		
	$\frac{T^o_{chol}}{T^s_{chol}}$	$\frac{T^o_{sol}}{T^s_{sol}}$	$\frac{T^o_{lp}}{T^s_{lp}}$	$\frac{T^m_{chol}}{T^s_{chol}}$	$\frac{T^m_{sol}}{T^s_{sol}}$	$\frac{T^m_{lp}}{T^s_{lp}}$
*ibmpg*3	12.4X	47.8X	16.3X	26.5X	30X	2.0X
*ibmpg*4	6.0X	26.2X	26.3X	12.9X	19.3X	4.4X
*ibmpg*6	3.0X	28.4X	16.0X	5.6X	6.7X	2.3X
*ibmpg*7	6.2X	30.5X	21.0X	13.7X	19.6X	2.1X
*thupg*1	8.3X	43.8X	1.1X	40.0X	24.1X	2.7X
*thupg*2	20.4X	93.8X	15.7X	53.7X	865X	2.5X

a single level (direct), multilevel method w/o and w/ sparsification methods, respectively; T^*_{chol}, T^*_{sol} and T^*_{lp} denote the runtime for Cholesky factorization, adjoint sensitivity calculation using matrix factors and total LP solve including all levels, respectively; Err denotes the relative error of maximum voltage drop compared to a single level method and κ denotes the relative condition number.

For all test cases, it is observed that matrix factorization and sensitivity calculation using the "Multilevel w/ Sparsification" method are the fastest among all three methods, while "Multilevel w/o Sparsification" is always the slowest due to the fast-growing matrix densities at coarse levels. The LP solution time T_{lp} with "Multilevel w/ Sparsification" is also the smallest since the number of variables and constraints in vectorless verification can be more significantly reduced on the sparsified grids than the original grid.

It is observed that solving LPs using the proposed method is over $2X$ faster than using the "Multilevel w/o Sparsification" method, and over $20X$ faster than using the

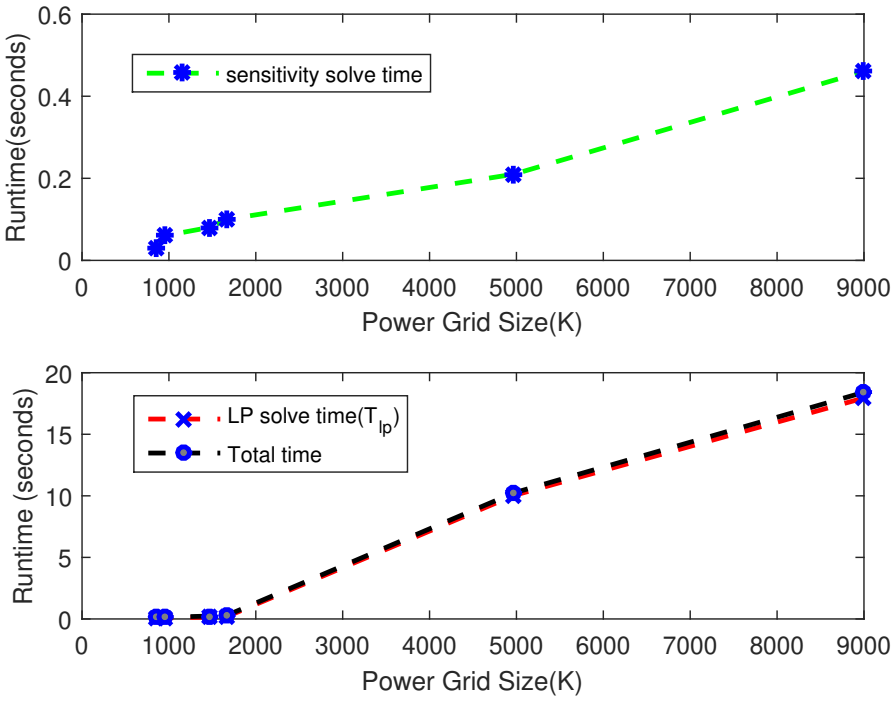


Figure 4.6: Runtime scalability of the proposed method.

"Single Level" method, showing that the proposed method can play a very important role in reducing overall computational cost during vectorless verifications, especially for large power grid designs. Figure 4.6 shows the nearly-linear runtime scalability of the proposed method, where both the LP solve time and adjoint sensitivity calculation time have been demonstrated.

4.3.2.3 Tradeoff Analysis Between Accuracy and Efficiency

Figure 4.7 shows how vectorless verification solution quality (error) and the LP runtime will change with different relative condition numbers (κ). As observed in our

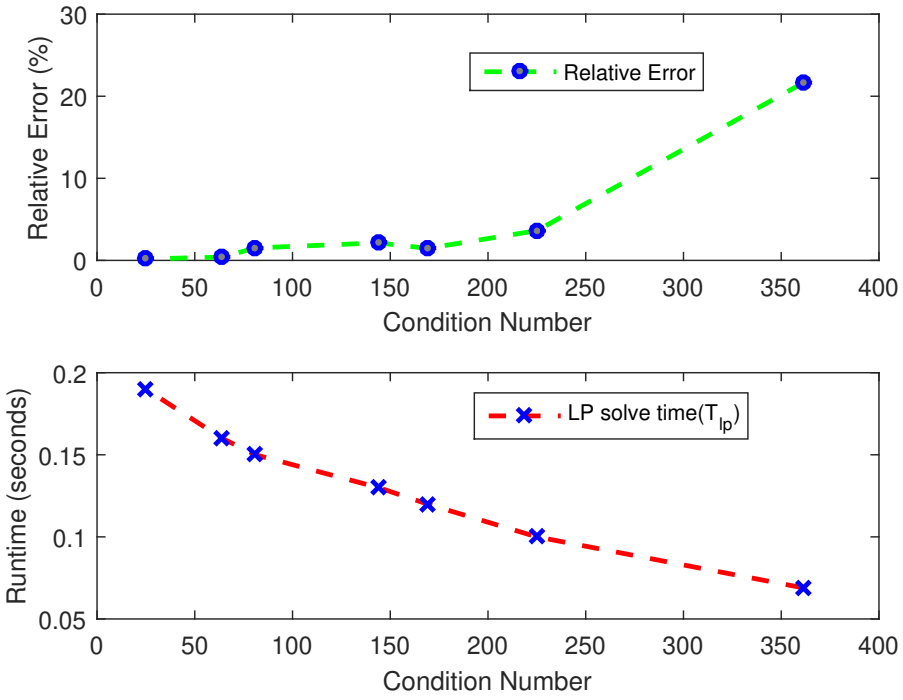


Figure 4.7: Result of the tradeoff analysis using the proposed method.

experimental results, the relative errors grow rather slowly with increasing condition numbers ($\kappa < 200$), while the LP solution time can be dramatically reduced. It can be shown that a larger condition number leads to a sparser power grid sparsifier with less number of current variables after node and constraint aggregations at coarse levels, thus faster verification procedure with worse approximations, which allows to flexibly explore the tradeoffs between the vectorless verification runtime and solution quality.

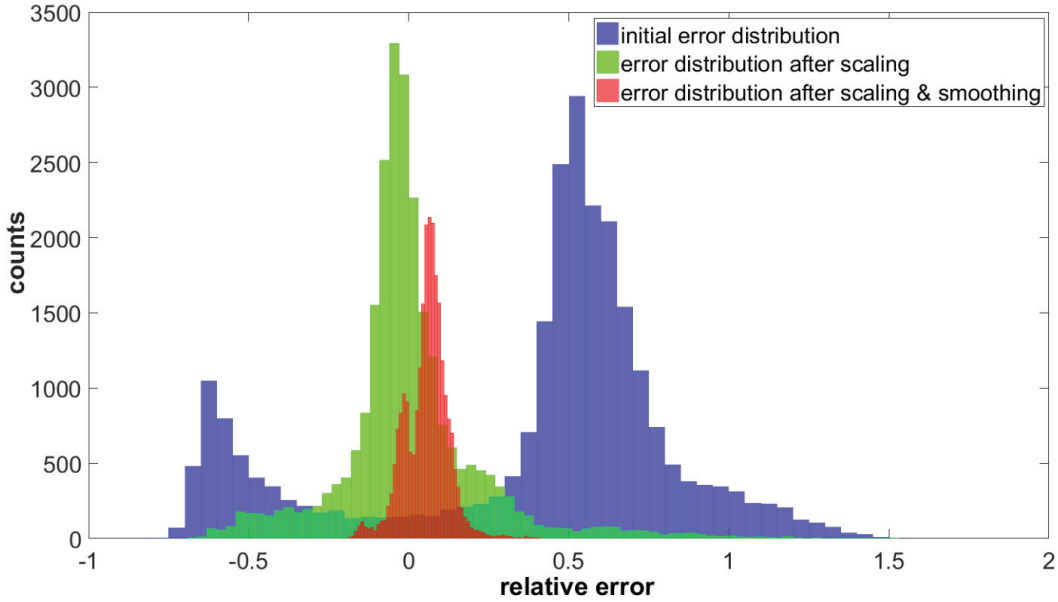


Figure 4.8: Relative Error Distributions.

4.3.3 Experimental Results for Thermal Verification

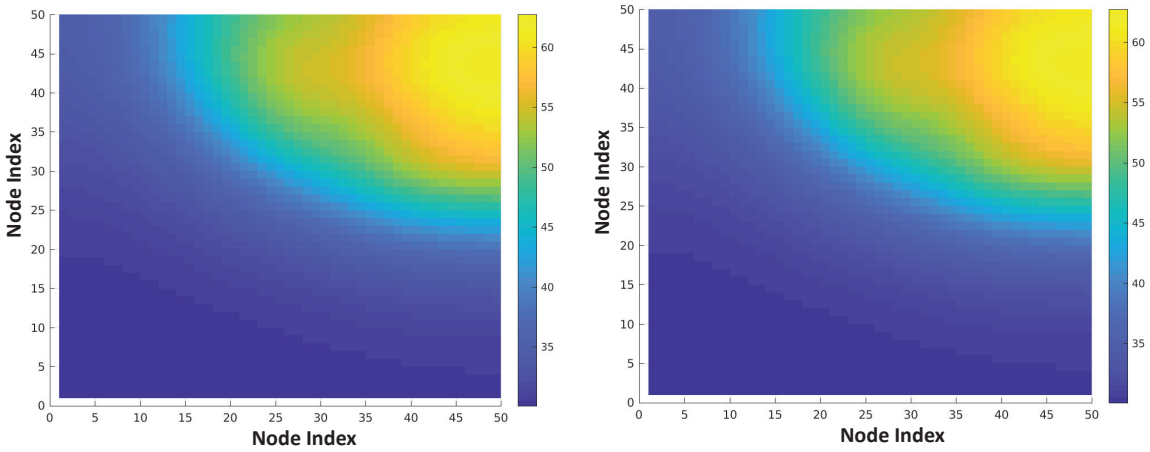
4.3.3.1 Iterative Edge Scaling and Solution Refinement

To demonstrate the effectiveness of the proposed edge scaling and solution refinement schemes, four solution (temperature) vectors are calculated for a 3D thermal grid and its spectral sparsifiers: **(a)** the true solution vector obtained using the original thermal grid, **(b)** the approximate solution vector computed using the sparsifier without edge scaling, **(c)** the approximate solution vector obtained using the sparsifier with edge scaling, **(d)** as well as the refined (smoothed) solution vector using the sparsifier with edge scaling. Meanwhile, we plot histogram distributions of relative errors of

the solution vectors (b)-(d) by comparing them against the true solution vector (a), as shown in Figure 4.8. We can see that the solution errors between the sparsifiers and the original graph can be significantly reduced by leveraging the proposed iterative edge scaling scheme and further mitigated by the proposed solution refinement procedure.

4.3.3.2 Result of Verification Quality

As we mentioned in the previous sections, the spectral graph sparsification method can well-preserve the low frequency components of the original thermal grid solutions, which will allow achieving high-quality solutions for vectorless verification tasks. Figures 4.9 and 4.10 show the worst-case thermal distributions of processors A and B using **(a)** the direct method, **(b)** the multilevel vectorless verification method w/o sparsification, and **(c)** the multilevel vectorless verification method w/ spectral sparsification, respectively. As observed, the three worst-case thermal distributions are very close to each other, indicating that rather accurate vectorless verification results can be obtained using spectrally-sparsified thermal grids.



(a) Thermal distribution by method (a)

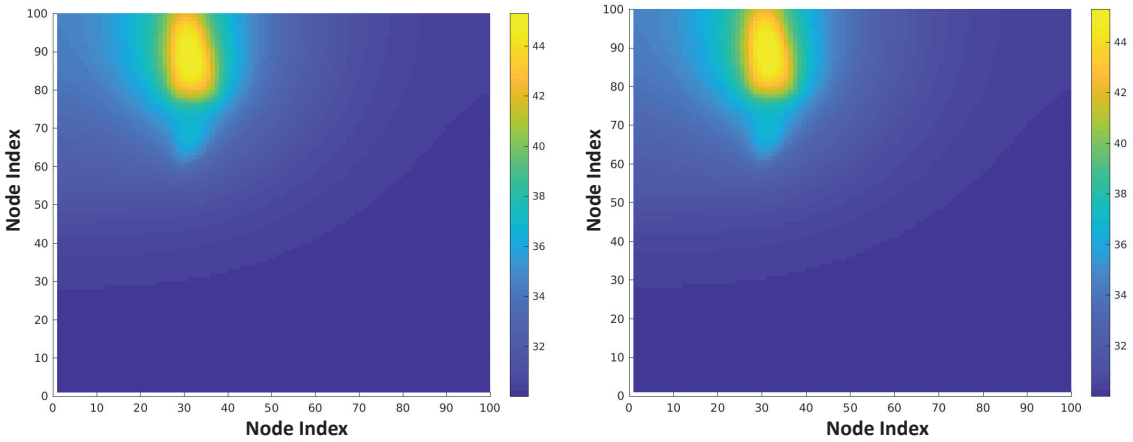
(b) Thermal distribution by method (b)

(c) Thermal distribution by method (c)

Figure 4.9: Worst-case temperature distributions of processor A

Table 4.5
Results of the proposed multilevel vectorless thermal integrity verification method (two-level scheme is used).

CKT	(a) Single Level			(b) Multilevel w/o Sparsification				(c) Multilevel w/ Sparsification				
	T^o_{chol}	T^o_{sol}	T^o_{lp}	T^m_{chol}	T^m_{sol}	T^m_{lp}	$Err(\%)$	T^s_{chol}	T^s_{sol}	T^s_{lp}	$Err(\%)$	κ
T1	$0.94s$	$2.30s$	$2.71s$	$1.24s$	$3.21s$	$3.12s$	1.0%	$0.03s$	$0.13s$	$2.02s$	5.0%	2,073
T2	$5.89s$	$14.79s$	$10.36s$	$8.12s$	$20.48s$	$5.80s$	2.1%	$0.29s$	$0.72s$	$6.81s$	3.8%	2,400
T3	$24.26s$	$55.20s$	$20.08s$	$33.91s$	$86.07s$	$25.90s$	4.0%	$1.13s$	$2.90s$	$4.33s$	4.0%	1,435
T4	$38.03s$	$99.91s$	$60.56s$	$50.91s$	$131.97s$	$24.15s$	2.0%	$4.61s$	$10.46s$	$14.48s$	5.0%	2,193
T5	$110.17s$	$262.53s$	$159.83s$	$148.11s$	$335.97s$	$21.43s$	1.0%	$20.09s$	$43.50s$	$9.36s$	2.0%	2,469
T6	$1.18Ks$	$33.60Ks$	$0.87Ks$	$1.25Ks$	$33.99Ks$	$0.79Ks$	1.0%	$51.70s$	$167.00s$	$133.83s$	1.0%	2,141
T7	$1.32Ks$	$32.27Ks$	$1.76Ks$	$1.42Ks$	$28.91Ks$	$1.70Ks$	1.0%	$65.23s$	$187.36s$	$181.76s$	2.0%	3,073



(a) Thermal distribution by method (a)

(b) Thermal distribution by method (b)

(c) Thermal distribution by method (c)

Figure 4.10: Worst-case temperature distributions of processor B

4.3.3.3 Comprehensive Results

Vectorless thermal integrity verification results using different methods are shown in Table 4.5. Except for T^*_{chol}, all other computational runtime are obtained by summing up the runtimes for verifying 100 randomly chosen nodes.

Table 4.6
Runtime results of the proposed method.

CKT	Sp. Over Single Level			Sp. Over Original Multilevel		
	$\frac{T^o_{chol}}{T^s_{chol}}$	$\frac{T^o_{sol}}{T^s_{sol}}$	$\frac{T^o_{lp}}{T^s_{lp}}$	$\frac{T^m_{chol}}{T^s_{chol}}$	$\frac{T^m_{sol}}{T^s_{sol}}$	$\frac{T^m_{lp}}{T^s_{lp}}$
$T1$	$31X$	$18X$	$1.4X$	$41X$	$25X$	$1.6X$
$T2$	$20X$	$20X$	$1.6X$	$28X$	$28X$	$0.85X$
$T3$	$22X$	$19X$	$4.6X$	$30X$	$30X$	$6.0X$
$T4$	$9X$	$10X$	$4X$	$11X$	$13X$	$1.7X$
$T5$	$6X$	$6X$	$18X$	$8X$	$8X$	$2.3X$
$T6$	$23X$	$201X$	$6.5X$	$24X$	$204X$	$5.9X$
$T7$	$20X$	$172X$	$9.7X$	$22X$	$154X$	$9.1X$

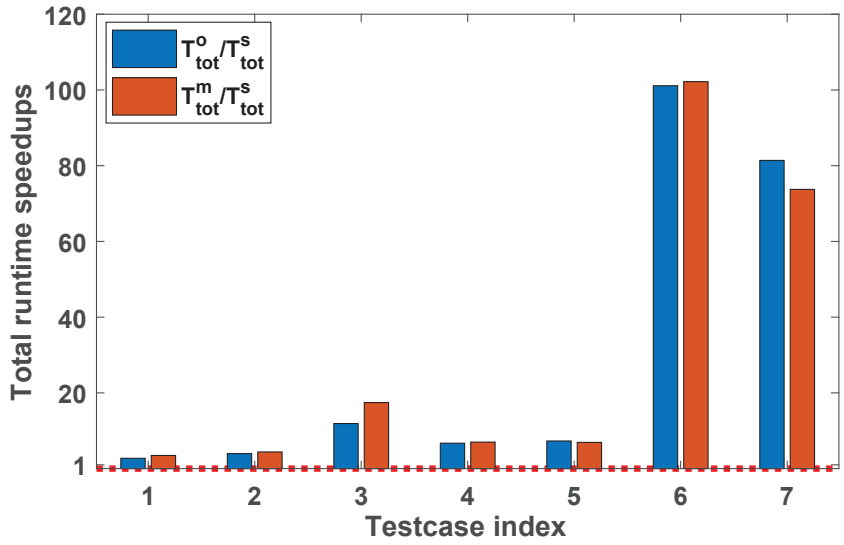
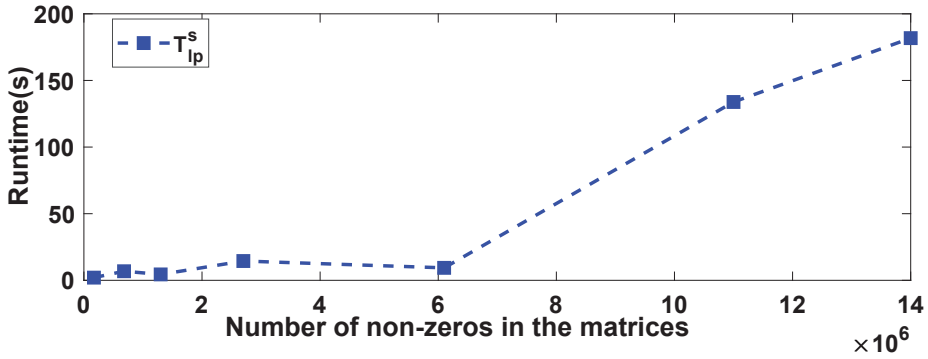


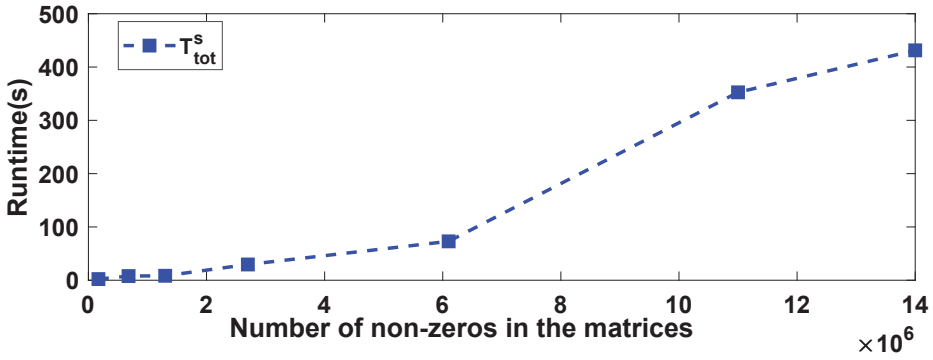
Figure 4.11: Total runtime speedups of Multilevel w/ Sparsification method comparing to the other two methods.

Table 4.6 shows the runtime speedups for Cholesky factorization, adjoint sensitivity calculation, and solving LP when comparing the proposed multilevel method with the other two methods, while Figure 4.11 shows the total runtime speedups of the proposed method compared to the other two methods. It is observed that both the matrix factorization and adjoint sensitivity calculation procedures in the "Multilevel w/ Sparsification" method are consistently much faster than the other two methods,

especially for larger test cases. And the total runtime speedups can be up to 100X for the larger thermal grids when using the proposed method. While the "Multilevel w/o Sparsification" is the slowest method due to the fast growing matrix densities at coarse levels. The overall LP solution time T_{lp} for the "Multilevel w/ Sparsification" method is also the smallest, indicating that the proposed method can effectively reduce the number of decision variables in LP and thus result in much lower computational cost in vectorless thermal verification tasks.



(a) LP solve time with the number of non-zero in matrices



(b) Total verification time with the number of non-zeros in matrices

Figure 4.12: Verification time with various problem sizes.

Meanwhile, the proposed method scales very comfortably with even very large 3D thermal grids, since both the LP solve time and total verification time increase almost

linearly with the 3D thermal grid sizes, as shown in Figure 4.12.

Chapter 5

Conclusion

This dissertation has presented algorithms and frameworks to tackle graph-related computing tasks by utilizing spectral methods. Three high-performance spectral methods for computer-aided design of integrated circuits are discussed in the dissertation:

1. We propose a scalable algorithmic framework for spectral reduction of large undirected graphs. The proposed method allows computing with much smaller graphs while preserving the key spectrum of the original graph. To achieve truly scalable performance (nearly-linear complexity) for spectral graph reduction, we leverage the recent similarity-aware spectral sparsification method, graph-theoretic algebraic

multigrid (AMG) Laplacian solver, and a novel constrained stochastic gradient descent (SGD) optimization approach in our spectral graph reduction algorithm.

We show that the resulting spectrally-reduced graphs can robustly preserve the first few nontrivial eigenvalues and eigenvectors of the original graph Laplacian. Besides, the spectral graph reduction method has been leveraged to develop much faster algorithms for multilevel spectral graph partitioning as well as t-distributed Stochastic Neighbor Embedding (t-SNE) of large data sets. We conducted extensive experiments using a variety of large graphs and data sets and obtained very promising results. For instance, we are able to reduce the "coPapersCiteseer" graph with $430K$ nodes and 16 million edges to a much smaller graph with only $13K$ (32X fewer) nodes and $17K$ (950X fewer) edges in about 16 seconds; the spectrally-reduced graphs also allow us to achieve up to $1100X$ speedup for spectral graph partitioning and up to $60X$ speedup for t-SNE visualization of large data sets. [18]

2. We introduce a Sparsified Algebraic Multigrid (SAMG) framework such that the scalability of recent graph-theoretic Algebraic Multigrid (AMG) solvers can be improved. We leverage a nearly-linear time spectral-perturbation based graph sparsification algorithm to aggressively sparsify the AMG coarse level problems without impacting the overall convergence of the solver. As a result, the coarse level problems generated by the proposed SAMG solver are always much sparser than the original problems without sacrificing spectral approximation quality, leading to scalable yet

robust AMG algorithms for solving large SDD matrices. Extensive experimental results show the proposed SAMG solver can significantly outperform the prior LAMG solver for a variety of large SDD matrix problems encountered in IC simulations, 3D-IC thermal analysis, image processing, and finite element analysis, as well as data mining and machine learning tasks.

3. We present a highly-scalable multilevel vectorless power grid and thermal integrity verification framework for computing chip worst-case voltage drop and thermal profiles without knowing the exact distribution of underlying power sources or workloads. Recent theoretical results in spectral graph sparsification and graph signal processing techniques enable us to develop much faster and more scalable vectorless thermal integrity verification algorithms while achieving flexible tradeoffs between computing efficiency and solution quality. Extensive experiment results for various chip designs have been demonstrated, indicating that the proposed scalable vectorless verification method can always efficiently obtain highly-accurate results for large chip designs.

5.1 Future Work

There are several research directions can be considered for the further extension of this dissertation:

1. The graph reduction framework can be further extended to the application of the circuit simulation. Even though the circuit networks contain nonlinear components, this framework can still be applied to linearized parts extracted from the circuit network, potentially reducing the complexity of the circuit simulation.

2. The SAMG solver can be potentially benefited by the proposed SGD edge scaling scheme. In the proposed SAMG algorithm, all edges are scaled up by the same scaling factor after spectral sparsification. However, the SGD edge scaling scheme achieves better sparsifier quality with a minimum number of edges, thus can be applied for better improving the graph approximation quality in SAMG solver.

3. In chapter 4, only static status (DC) is considered for power grid and thermal integrity verification. However, the framework of the verification can also apply to integrated circuits for transient simulations.

References

[1] C. J. Alpert. The ispd98 circuit benchmark suite. In *Proceedings of the 1998 international symposium on Physical design*, pages 80–85. ACM, 1998.

[2] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

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

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

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

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

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

[8] W. L. Briggs, S. F. McCormick, et al. *A multigrid tutorial*, volume 72. Siam, 2000.

[9] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz. Recent advances in graph partitioning. In *Algorithm Engineering*, pages 117–158. Springer, 2016.

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

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

[12] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in

undirected graphs. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 273–282, 2011.

[13] R. Cochran and S. Reda. Consistent runtime thermal prediction and control through workload phase detection. In *Proceedings of the 47th Design Automation Conference*, pages 62–67. ACM, 2010.

[14] M. B. Cohen, J. Kelner, J. Peebles, R. Peng, A. B. Rao, A. Sidford, and A. Vladu. 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*, pages 410–419. ACM, 2017.

[15] D. Kouroussis and I. Ferzli and F. Najm. Incremental partitioning-based vectorless power grid verification. In *Proc. IEEE/ACM ICCAD*, pages 358–364, 2005.

[16] T. Davis. *CHOLMOD: sparse supernodal Cholesky factorization and update/downdate*. [Online]. Available: http://www.cise.ufl.edu/research/sparse/cholmod/, 2008.

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

[18] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[19] C. Deng, Z. Zhao, Y. Wang, Z. Zhang, and Z. Feng. Graphzoom: A multi-level spectral approach for accurate and scalable graph embedding. In *International Conference on Learning Representations*, 2020.

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

[21] W. Donath and A. Hoffman. Algorithms for partitioning of graphs and computer logic based on eigenvectors of connections matrices. *IBM Technical Disclosure Bulletin*, 15, 1972.

[22] W. E. Donath and A. J. Hoffman. Lower bounds for the partitioning of graphs. In *Selected Papers Of Alan J Hoffman: With Commentary*, pages 437–442. World Scientific, 2003.

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

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

[25] Z. Feng. Scalable multilevel vectorless power grid voltage integrity verification. *IEEE Trans. on VLSI Systems*, 21(8):1526–1539, August 2013.

[26] Z. Feng. Scalable Vectorless Power Grid Current Integrity Verification. In *Proc. of IEEE/ACM DAC*, pages 86:1–86:8, 2013.

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

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

[29] Z. Feng. Grass: Graph spectral sparsification leveraging scalable spectral perturbation analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[30] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

[31] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25(4):619–633, 1975.

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

[33] N. Ghani and F. Najm. Fast Vectorless Power Grid Verification Using an Approximate Inverse Technique. In *Proc. IEEE/ACM DAC*, pages 184–189, 2009.

[34] A. Goyal and F. N. Najm. Efficient rc power grid verification using node elimination. In *Proc. of IEEE/ACM DATE*, pages 257–260, 2011.

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

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

[37] R. A. Horn, R. A. Horn, and C. R. Johnson. *Matrix analysis*. Cambridge university press, 1990.

[38] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan. Hotspot: A compact thermal modeling methodology for early-stage vlsi design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(5):501–513, 2006.

[39] M. Imre, J. Tao, Y. Wang, Z. Zhao, Z. Feng, and C. Wang. Spectrum-preserving sparsification for visualization of big graphs. *Computers & Graphics*, 87:89–102, 2020.

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

[41] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: applications in vlsi domain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 7(1):69–79, 1999.

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

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

[44] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. *VLSI design*, 11(3):285–300, 2000.

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

[46] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970.

[47] P. Kolev and K. Mehlhorn. A note on spectral clustering. *arXiv preprint arXiv:1509.09188*, 2015.

[48] A. Kolla, Y. Makarychev, A. Saberi, and S. Teng. Subgraph sparsification and nearly optimal ultrasparsifiers. In *Proc. ACM STOC*, pages 57–66, 2010.

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

[50] D. Kouroussis and F. Najm. A static pattern-independent technique for power grid voltage integrity verification. In *Proc. IEEE/ACM DAC*, pages 99–104, 2003.

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

[52] I. Koutis, G. Miller, and D. Tolliver. Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing. *Computer Vision and Image Understanding*, 115(12):1638–1646, 2011.

[53] I. Koutis, G. L. Miller, and R. Peng. Approaching optimality for solving sdd linear systems. *SIAM Journal on Computing*, 43(1):337–354, 2014.

[54] D. LaSalle, M. M. A. Patwary, N. Satish, N. Sundaram, P. Dubey, and G. Karypis. Improving graph partitioning for modern graphs and architectures. In *Proceedings of the 5th Workshop on Irregular Applications: Architectures and Algorithms*, page 14. ACM, 2015.

[55] J. R. Lee, S. O. Gharan, and L. Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM (JACM)*, 61(6):37, 2014.

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

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

[58] R. Lewis and S. Nash. Model problems for the multigrid optimization of systems governed by differential equations. *SIAM J. Sci. Comput.*, 26(6):1811–1837, 2005.

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

[60] P. Li, L. T. Pileggi, M. Asheghi, and R. Chandra. Ic thermal simulation and modeling via efficient multigrid-based approaches. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(9):1763–1776, 2006.

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

[62] M. P.-H. Lin, H. Zhang, M. D. Wong, and Y.-W. Chang. Thermal-driven analog placement considering device matching. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(3):325–336, 2011.

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

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

[65] A. Loukas. Graph reduction with spectral and cut guarantees. *arXiv preprint arXiv:1808.10650*, 2018.

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

[67] A. Loukas and P. Vandergheynst. Spectrally approximating large graphs with smaller graphs. In *International Conference on Machine Learning*, pages 3237–3246, 2018.

[68] A. Loukas and P. Vandergheynst. Spectrally approximating large graphs with smaller graphs. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 3237–3246, 2018.

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

[70] F. Najm. Overview of vectorless/early power grid verification. In *Proc. of IEEE/ACM ICCAD*, pages 670–677, 2012.

[71] S. R. Nassif. *IBM power grid benchmarks*. [Online]. Available: http://dropzone.tamu.edu/ pli/PGBench/, 2008.

[72] M. Naumov and T. Moon. Parallel spectral graph partitioning. Technical report, NVIDIA Technical Report, NVR-2016-001, 2016.

[73] M. E. Newman. Community detection and graph partitioning. *EPL (Europhysics Letters)*, 103(2):28003, 2013.

[74] G. Optimization. Gurobi optimizer [online]. available: www. gurobi. com, 2016.

[75] G. A. Pavlopoulos, M. Secrier, C. N. Moschopoulos, T. G. Soldatos, S. Kossida, J. Aerts, R. Schneider, and P. G. Bagos. Using graph theory to analyze biological networks. *BioData mining*, 4(1):10, 2011.

[76] M. Pedram and S. Nazarian. Thermal modeling, analysis, and management in vlsi circuits: Principles and methods. *Proceedings of the IEEE*, 94(8):1487–1501, 2006.

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

[78] R. Peng, H. Sun, and L. Zanetti. Partitioning well-clustered graphs: Spectral clustering works! In *Conference on Learning Theory*, pages 1423–1455, 2015.

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

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

[81] H. Qian, S. R. Nassif, and S. S. Sapatnekar. Early-stage power grid analysis for uncertain working modes. *IEEE Trans. on Computer-Aided Design*, 24(5):676–682, 2005.

[82] J. Ruge and K. Stüben. Algebraic multigrid. *Multigrid methods*, 3(13):73–130, 1987.

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

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

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

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

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

[88] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan. Temperature-aware microarchitecture: Modeling and implementation. *ACM Transactions on Architecture and Code Optimization (TACO)*, 1(1):94–125, 2004.

[89] D. Spielman. Algorithms, graph theory, and linear equations in Laplacian matrices. In *Proceedings of the International Congress of Mathematicians (ICM)*, volume 4, pages 2698–2722, 2010.

[90] D. Spielman and N. Srivastava. Graph sparsification by effective resistances. In *Proc. ACM STOC*, pages 563–568, 2008.

[91] D. Spielman and S. Teng. Nearly-linear time algorithms for graph partitioning,

graph sparsification, and solving linear systems. In *Proc. ACM STOC*, pages 81–90, 2004.

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

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

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

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

[96] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

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

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

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

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

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

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

[103] Y. Wang, Z. Zhao, and Z. Feng. Graspel: Graph spectral learning at scale. *arXiv preprint arXiv:1911.10373*, 2019.

[104] Y.-C. Wei and C.-K. Cheng. Ratio cut partitioning for hierarchical designs. *IEEE Transactions on computer-aided design*, 10(7):911–921, 1991.

[105] X. Xiong and J. Wang. An Efficient Dual Algorithm for Vectorless Power Grid Verification under Linear Current Constraints. In *Proc. IEEE/ACM DAC*, pages 837–842, 2010.

[106] F. Xue. *Numerical solution of eigenvalue problems with spectral transformations.* PhD thesis, University of Maryland, College Park, 2009.

[107] J. Yang, Y. Cai, Q. Zhou, and W. Zhao. A selected inversion approach for locality driven vectorless power grid verification. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(11):2617–2628, 2015.

[108] J. Yang and Z. Li. *THU power grid benchmarks*. [Online]. Available: http://tiger.cs.tsinghua.edu.cn/PGBench/.

[109] Y. Zhang, Z. Zhao, and Z. Feng. Towards scalable spectral sparsification of directed graphs. In *2019 IEEE International Conference on Embedded Software and Systems (ICESS)*, pages 1–2. IEEE, 2019.

[110] Y. Zhang, Z. Zhao, and Z. Feng. SF-GRASS: Solver-Free Graph Spectral Sparsification. In *Proceedings of ACM/IEEE International Conference on Computer-Aided Design*, 2020.

[111] Y. Zhang, Z. Zhao, and Z. Feng. A unified approach to scalable spectral sparsification of directed graphs. *arXiv preprint arXiv:1812.04165*, 2020.

[112] X. Zhao, Z. Feng, and C. Zhuo. An efficient spectral graph sparsification approach to scalable reduction of large flip-chip power grids. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 218–223. IEEE, 2014.

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

[114] Z. Zhao and Z. Feng. Effective-resistance preserving spectral reduction of graphs. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[115] Z. Zhao and Z. Feng. A spectral approach to scalable vectorless thermal integrity verification. In *2020 Design, Automation & Test in Europe*. IEEE, 2020.

[116] 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.

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

[118] Z. Zhao, Y. Zhang, and Z. Feng. Towards scalable spectral embedding and data visualization via spectral coarsening. In *14th ACM International Conference on Web Searching and Data Mining (WSDM) (to be appear)*, 2021.

[119] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. of ICML*, volume 3, pages 912–919, 2003.

Appendix A

Supplementary Materials

A.1 Spectral Graph Partitioning

Graph partitioning is one of the fundamental algorithmic operations, which can be applied to many fields [9], such as parallel processing, community detection in social networks [73], biological networks analysis [75], VLSI computer-aided design [42], etc. The graph partitioning aims to partition the vertices or edges of a graph into a number of disjoint sets without introducing too many connections between the sets. A variety of graph partitioning algorithms has been proposed, from local heuristics like Kernighan-Lin [46] to global methods such as spectral partitioning [9, 31] and multilevel partitioning [43]. Spectral partitioning, which was first noted in [21, 22],

30, 31], has become one of the most important methods for graph partitioning.

Consider a weighted graph $G = (V, E_G, w_G)$ with vertex (node) set $V = \{v_1, \cdots, v_n\}$ denoting n vertices in the graph, edge set E_G representing weighted edges in the graph and w_G denoting a weight function that assigns positive weight to all edges, that is $w_G(p, q) > 0$ if there is an edge connecting node v_p and node v_q, which can also be represented by $(p, q) \in E_G$. Given a subset of vertices $S \subset V$ and its complement set $\bar{S} = V \backslash S$, the boundary between set S and set $\bar{S}$ is defined as a set of edges $B(S, \bar{S}) \subset E_G$ such that one node of each edge is in set S and the other node is in set $\bar{S}$:

$$B(S) = \{(p, q) : p \in S \wedge q \in \bar{S}\}. \tag{A.1}$$

The cut between S and $\bar{S}$ can be defined as follows:

$$C(S, \bar{S}) = \sum_{(p,q) \in B(S)} w_G(p, q) = \text{vol}(B(S)). \tag{A.2}$$

The simplest idea of graph partitioning is to find a partition of the graph so that different partition sets are weakly connected (meaning the edges between different sets have low weights) while the interior of each set is strongly connected. The aim of graph partitioning is to find the set S such that $C(S, \bar{S})$ can be minimized. However, in practice the solution of this min-cut problem is usually unacceptable due to the highly unbalanced partitioning results. For example, the resulted set S may only have one individual vertex while $\bar{S}$ includes rest of the graph. Therefore, we also want the

partitions to be reasonably balanced. To realize the minimum balanced cut of graph partitioning, two objective functions have been introduced: ratio cut $\rho(S)$ [104] and normalized cut $\theta(S)$ [85], which have been defined as follows:

$$\rho(S) = \min_S \frac{|V|C(S,\bar{S})}{|S||\bar{S}|} = \min_S \left(\frac{C(S,\bar{S})}{|S|} + \frac{C(S,\bar{S})}{|\bar{S}|} \right) \tag{A.3}$$

$$\theta(S) = \min_S \frac{\text{vol}(V)C(S,\bar{S})}{\text{vol}(S)\text{vol}(\bar{S})} = \min_S \left(\frac{C(S,\bar{S})}{\text{vol}(S)} + \frac{C(S,\bar{S})}{\text{vol}(\bar{S})} \right) \tag{A.4}$$

where

$$|S| := \text{the number of vertices in set S} \tag{A.5}$$

$$\text{vol}(S) = \sum_{p \in S \wedge (p,q) \in E_G} w_G(p,q). \tag{A.6}$$

Note that number of vertices (sum of edge weights) is used to measure the size of set S for ratio cut $\rho(S)$ (normalized cut $\theta(S)$). In other words, the ratio cut $\rho(S)$ metric aims to balance the number of vertices for each set, while the normalized cut $\theta(S)$ metric aims to balance number of edges in each set. The ratio cut in (A.3) and normalized cut in (A.4) can be generalized as follows for k-way partitioning problems

[72, 100]:

$$\rho(S_1, \cdots, S_k) = \min_{S_1,\cdots,S_k} \sum_{i=1}^{k} \frac{C(S_i, \bar{S}_i)}{|S_i|} \tag{A.7}$$

$$\theta(S_1, \cdots, S_k) = \min_{S_1,\cdots,S_k} \sum_{i=1}^{k} \frac{C(S_i, \bar{S}_i)}{\text{vol}(S_i)}, \tag{A.8}$$

while the edge cut of k partitions becomes

$$C(S_1, \cdots, S_k) = \sum_{i=1}^{k} C(S_i, \bar{S}_i). \tag{A.9}$$

Since the optimization problems of (A.7) and (A.8) are NP-hard, spectral partitioning methods have been proposed for solving the relaxed optimization problems. It can be shown that the solution of the relaxed optimization problem (A.7) is the matrix of U with first k eigenvectors of the graph Laplacian as its columns vectors, whereas the solution to the relaxed optimization problem (A.8) is the matrix of U with the first k eigenvectors of the normalized graph Laplacian [85]. Detailed proof can be found in Section A.1.1 and Section A.1.2.

Since we want to partition V into k sets based on the indicator matrix $U \in \mathbb{R}^{nxk}$, one straightforward way is to treat each row of the matrix U as a point in a k dimensional space and use clustering algorithms, like k-means [2] to identify the k partitions.

A.1.1 Ratio cut and normalized cut for 2-way partitioning

Given the graph $G(V, E_G, w_G)$, the graph Laplacian L_G is defined as follows:

$$\mathbf{L}_G(i,j) = \begin{cases} -w_G(i,j) & \text{if } (i,j) \in E_G \\ \sum\limits_{(i,t)\in E_G} w_G(i,t) & \text{if } (i=j) \\ 0 & \text{if } otherwise. \end{cases} \tag{A.10}$$

L_G can also be represented as

$$\mathbf{L_G} = \mathbf{D_G} - \mathbf{A_G}, \tag{A.11}$$

where $\mathbf{A}_G$ is the adjacency matrix of the graph and $\mathbf{D}_G$ is the diagonal matrix with each i-th diagonal element being the sum of all elements in that row of $\mathbf{A}_G$. To relate the ratio cut objective function with the unnormalized graph Laplacian, we first define the vector $\mathbf{z} = (z_1, z_2, \cdots, z_n)^T \in \mathbb{R}^n$ with entries noted as follows [72, 100]:

$$z_i = \begin{cases} \sqrt{(|\bar{S}|/|S|)} & \text{if } v_i \in S \\ -\sqrt{(|S|/|\bar{S}|)} & \text{if } v_i \in \bar{S} \end{cases} \tag{A.12}$$

Then we have

$$\begin{aligned}\mathbf{z^T L_G z} &= \frac{1}{2}\sum_{p,q=1}^{n} w_G(p,q)(z_p - z_q)^2 \\ &= C(S,\bar{S})\left(\frac{|S|+|\bar{S}|}{|S|} + \frac{|S|+|\bar{S}|}{|\bar{S}|}\right) \\ &= |V| \cdot \frac{|V|C(S,\bar{S})}{|S||\bar{S}|} = |V|\rho(S)\end{aligned}$$

Given the all-one vector $\mathbf{1}$, the following can be observed:

$$\mathbf{z}^T\mathbf{1} = \sum_{i=1}^{n} z_i = \sum_{i\in S}\sqrt{\frac{|\bar{S}|}{|S|}} - \sum_{i\in\bar{S}}\sqrt{\frac{|S|}{|\bar{S}|}} = 0, \tag{A.13}$$

$$\mathbf{z^T z} = \sum_{i=1}^{n} z_i^2 = \sum_{i\in S}\frac{|\bar{S}|}{|S|} + \sum_{i\in\bar{S}}\frac{|S|}{|\bar{S}|} = n = |V|, \tag{A.14}$$

which will lead to:

$$\rho(S) = \frac{\mathbf{z^T L_G z}}{\mathbf{z^T z}}. \tag{A.15}$$

Since the values of z_i are restricted to the two particular values, this optimization problem is NP-hard. Spectral partitioning relaxes the constraints and allows $\mathbf{z}$ to take any real values.

According to the **Courant-Fischer theorem** [37], the solution to the relaxed optimization problem is the eigenvector of $\mathbf{L_G}$ associated with the smallest non-zero eigenvalue. Once the solution vector $\mathbf{z}$ is computed, a partition can be obtained by

converting the real-valued vector $\mathbf{z}$ to a discrete vector containing only 0 and 1 as the indicators for partitioning purpose. For example, one simple way is to use the sign of the vector $\mathbf{z}$ to partition the graph so that $v_i \in S$ if $z_i > 0$, otherwise $v_i \in \bar{S}$. Similar analysis can be performed for normalized cut by setting the vector $\mathbf{z}$ to be:

$$z_i = \begin{cases} \sqrt{(\text{vol}(\bar{S})/\text{vol}(S))} & \text{if } v_i \in S \\ -\sqrt{(\text{vol}(S)/\text{vol}(\bar{S}))} & \text{if } v_i \in \bar{S} \end{cases} \tag{A.16}$$

which leads to:

$$\mathbf{z^T L_G z} = \text{vol}(V)\theta(S) \tag{A.17}$$

$$(\mathbf{D_G z})^\mathbf{T} 1 = 0 \tag{A.18}$$

$$\mathbf{z^T D_G z} = \text{vol}(V) \tag{A.19}$$

By relaxing the vector $\mathbf{z}$ to take arbitrary real values, we can show that the solution to this relaxed optimization problem is the eigenvector associated to the second smallest eigenvalue to the generalized eigenvalue problem of

$$\mathbf{L_G u} = \lambda \mathbf{D_G u}. \tag{A.20}$$

A.1.2 Ratio cut and normalized cut for k-way partitioning

We follow a similar discussion based on previous analysis when relaxing the ratio cut and normalized cut minimizations, generalizing to k partitions. Given a partition of V into k sets, we define k indicator vectors $\mathbf{m_j} = (m_{1,j}, \cdots, m_{n,j})^T$ with $j = 1, \cdots, k$ such that

$$m_{i,j} = \begin{cases} 1/\sqrt{|S_j|} & \text{if } v_i \in S_j \\ 0 & \text{otherwise} \end{cases} \tag{A.21}$$

where $i = 1, \cdots, n; \;\; j = 1, \cdots, k$.

The indicator matrix $\mathbf{U}$ can be defined with the k vectors so that $\mathbf{U} = [\mathbf{m_1}, \cdots, \mathbf{m_k}]$. Note that columns in $\mathbf{U}$ are orthogonal to each other, that is $\mathbf{U^T U} = \mathbf{I}$. We also note that

$$\mathbf{m_j^T L_G m_j} = (\mathbf{U^T L_G U})_{jj} = \frac{C(S_j, \bar{S}_j)}{|S_j|} \tag{A.22}$$

By substituting (A.22) to (A.7) we can get

$$\rho(S_1, \cdots, S_k) = \sum_{j=1}^{k} \mathbf{m_j^T L_G m_j} = \sum_{j=1}^{k} (\mathbf{U^T L_G U})_{jj} = Tr(\mathbf{U^T L_G U}) \tag{A.23}$$

Where $Tr(\cdot)$ is the trace of a matrix. By relaxing the entries of indicator matrix $\mathbf{U}$ to be arbitrary real values, the optimization problem in (A.7) becomes

$$\rho(S) = \min_{\mathbf{U} \in \mathbb{R}^{n \times k}} Tr(\mathbf{U^T L_G U}) \quad \text{subject to: } \mathbf{U^T U} = \mathbf{I}.$$

According to the **Courant-Fischer** theorem, the solution to this optimization problem is the matrix of $\mathbf{U}$ with first k eigenvectors of $\mathbf{L_G}$ as its columns.

Similarly, we can choose the entries of indicator matrix $\mathbf{U}$ as follows:

$$m_{i,j} = \begin{cases} 1/\sqrt{\text{vol}(S_j)} & \text{if } v_i \in S_j \\ 0 & \text{otherwise .} \end{cases} \tag{A.24}$$

We observe that $\mathbf{U^T D_G U} = \mathbf{I}$, and $\mathbf{m_j^T L_G m_j} = C(S_j, \bar{S}_j)/\text{vol}(S_j)$. By relaxing $\mathbf{U}$ to take arbitrary real values, we can reformulate the optimization problem in (A.8) as

$$\theta(S) = \min_{\mathbf{U} \in \mathbb{R}^{n \times k}} Tr(\mathbf{U^T L_G U}) \quad \text{subject to: } \mathbf{U^T D_G U} = \mathbf{I}.$$

According to the **Courant-Fischer** theorem, the solution to this optimization problem is the matrix of $\mathbf{U}$ with first k generalized eigenvectors of $\mathbf{L_G u} = \lambda \mathbf{D_G u}$ as its columns [85].

A.2 t-Distributed Stochastic Neighbor Embedding

t-Distributed Stochastic Neighbor Embedding (t-SNE) [69, 99] is a nonlinear dimensionality reduction method designed for visualizing data. The goal of t-SNE is to learn a mapping from the high-dimensional space to a low-dimensional space with desired number of dimensionality in such a way that similar data points are mapped to nearby locations and dissimilar data points are mapped to distant locations. To accomplish this, t-SNE converts the Euclidean distances between data points in high-dimensional space into conditional probability as follows:

$$P_{j|i} = \frac{exp(-\frac{\|x_i - x_j\|^2}{2{\sigma_i}^2})}{\sum_{k \neq i} exp(-\frac{\|x_i - x_k\|^2}{2{\sigma_i}^2})}, \quad P_{i|i} = 0, \tag{A.25}$$

where σ_i denotes the variance of the Gaussian distribution that is centered at x_i. The joint probability is defined by symmetrizing a pair of conditional probabilities as

follows:

$$P_{ij} = \frac{P_{j|i} + P_{i|j}}{2N}, \tag{A.26}$$

t-SNE uses this joint probability to measure the similarity between two data points x_i and x_j in high-dimensional space. Denoting the corresponding points in low-dimensional space by y_i and y_j, respectively, then the similarity between them is measured by the following joint probability using the Cauchy kernel:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}}, \quad q_{ii} = 0. \tag{A.27}$$

t-SNE uses Kullback-Leibler (KL) divergence to measure the faithfulness of the embedding. The cost function is defined as the sum of KL divergence over all pairs of data points in the data set:

$$C = KL(P \parallel Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{A.28}$$

The embedding points in low-dimensional space $\{y_1, ..., y_N\}$ are determined by minimizing the KL cost function. Typically, starting with a random initialization, the cost function is minimized using gradient descent method with the following gradient:

$$\frac{\partial C}{\partial y_i} = 4 \sum_{i \neq j} (p_{ij} - q_{ij}) q_{ij} Z (y_i - y_j), \tag{A.29}$$

where the constant normalization term Z is given by:

$$Z = \sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}. \tag{A.30}$$

It should be noted that as the size of data set grows, the convergence rate will usually slow down [69, 99]. Computing gradients is very time-consuming, since it is an N-body problem that has a complexity of $O(N^2)$. By splitting the gradient into two parts, we have:

$$\begin{aligned} \frac{\partial C}{\partial y_i} &= 4 \sum_{i \neq j} p_{ij} q_{ij} Z(y_i - y_j) - 4 \sum_{i \neq j} q_{ij}^2 Z(y_i - y_j) \\ &= 4F_{attr,i} - 4F_{rep,i}, \end{aligned} \tag{A.31}$$

where $F_{attr,i}$ denotes the sum of attractive force acting on data point i and the $F_{rep,i}$ denotes the sum of repulsive force acting on data point i. Both forces are due to the rest of the data points. The position of each data point after embedding is determined by the net force acting on it.

In recent years, due to the prevalence of high-dimensional data, the t-SNE algorithm has become the most effective visualization tool due to its capability of performing dimensionality reduction in such a way that similar data points in high-dimensional space are embedded to nearby locations in the low-dimensional space of two or three dimensions with high probability. However, t-SNE is limited in its applicability to large real-world data sets due to the high computational complexity. In practice, the

standard t-SNE can not even apply to data sets with more than $10,000$ data points [69]. Thus, there is a pressing need to develop acceleration techniques for the t-SNE algorithm that can be adapted to visualize large-scale data sets. In the past decade, substantial effort has been made to reduce the computational cost of t-SNE. For example, tree-based algorithms have been proposed to accelerate the computation of the gradient in t-SNE [99], which, however, has no theoretical guarantee of the solution quality.

Appendix B

Copyright Permission

ACM Copyright and Audio/Video Release

Title of the Work: Effective-Resistance Preserving Spectral Reduction of Graphs

Author/Presenter(s): Zhiqiang Zhao;Zhuo Feng
Type of material: Full Paper

Publication and/or Conference Name: DAC '19: The 56th Annual Design Automation Conference 2019 Proceedings

I. Copyright Transfer, Reserved Rights and Permitted Uses

* Your Copyright Transfer is conditional upon you agreeing to the terms set out below.

Copyright to the Work and to any supplemental files integral to the Work which are submitted with it for review and publication such as an extended proof, a PowerPoint outline, or appendices that may exceed a printed page limit, (including without limitation, the right to publish the Work in whole or in part in any and all forms of media, now or hereafter known) is hereby transferred to the ACM (for Government work, to the extent transferable) effective as of the date of this agreement, on the understanding that the Work has been accepted for publication by ACM.

Reserved Rights and Permitted Uses

(a) All rights and permissions the author has not granted to ACM are reserved to the Owner, including all other proprietary rights such as patent or trademark rights.

(b) Furthermore, notwithstanding the exclusive rights the Owner has granted to ACM, Owner shall have the right to do the following:

(i) Reuse any portion of the Work, without fee, in any future works written or edited by the Author, including books, lectures and presentations in any and all media.

(ii) Create a "Major Revision" which is wholly owned by the author

(iii) Post the Accepted Version of the Work on (1) the Author's home page, (2) the Owner's institutional repository, (3) any repository legally mandated by an agency funding the research on which the Work is based, and (4) any non-commercial repository or aggregation that does not duplicate ACM tables of contents, i.e., whose patterns of links do not substantially duplicate an ACM-copyrighted volume or issue. Non-commercial repositories are here understood as repositories owned by non-profit organizations that do not charge a fee for accessing deposited articles and that do not sell advertising or otherwise profit from serving articles.

(iv) Post an "Author-Izer" link enabling free downloads of the Version of Record in the ACM Digital Library on (1) the Author's home page or (2) the Owner's institutional repository;

(v) Prior to commencement of the ACM peer review process, post the version of the Work as submitted to ACM ("Submitted Version" or any earlier versions) to non-peer reviewed servers;

(vi) Make free distributions of the final published Version of Record internally to the Owner's employees, if applicable;

(vii) Make free distributions of the published Version of Record for Classroom and Personal Use;

(viii) Bundle the Work in any of Owner's software distributions; and

(ix) Use any Auxiliary Material independent from the Work. (x) If your paper is withdrawn before it is published in the ACM Digital Library, the rights revert back to the author(s).

https://doi.org/10.1145/3316781.3317809

NOTE: Make sure to include your article's DOI as part of the bibstrip data; DOIs will be registered and become active shortly after publication in the ACM Digital Library. Once you have your camera ready copy ready, please send your source files and PDF to your event contact for processing.

☑ A. Assent to Assignment. I hereby represent and warrant that I am the sole owner (or authorized agent of the copyright owner(s)), with the exception of third party materials detailed in section III below. I have obtained permission for any third-party material included in the Work.

☐ B. Declaration for Government Work. I am an employee of the National Government of my country and my Government claims rights to this work, or it is not copyrightable (Government work is classified as Public Domain in U.S. only)

II. Permission For Conference Recording and Distribution

* Your Audio/Video Release is conditional upon you agreeing to the terms set out below.

I hereby grant permission for ACM to include my name, likeness, presentation and comments in any and all forms, for the Conference and/or Publication.

I further grant permission for ACM to record and/or transcribe and reproduce my presentation as part of the ACM Digital Library, and to distribute the same for sale in complete or partial form as part of an ACM product on CD-ROM, DVD, webcast, USB device, streaming video or any other media format now or hereafter known.

I understand that my presentation will not be sold separately as a stand-alone product without my direct consent. Accordingly, I give ACM the right to use my image, voice, pronouncements, likeness, and my name, and any biographical material submitted by me, in connection with the Conference and/or Publication, whether used in excerpts or in full, for distribution described above and for any associated advertising or exhibition.

Do you agree to the above Audio/Video Release? ◉ Yes ○ No

III. Auxiliary Material

Do you have any Auxiliary Materials? ○ Yes ◉ No

IV. Third Party Materials

In the event that any materials used in my presentation or Auxiliary Materials contain the work of third-party individuals or organizations (including copyrighted music or movie excerpts or anything not owned by me), I understand that it is my responsibility to secure any necessary permissions and/or licenses for print and/or digital publication, and cite or attach them below.

◉ We/I have not used third-party material.
○ We/I have used third-party materials and have necessary permissions.

V. Artistic Images

If your paper includes images that were created for any purpose other than this paper and to which you or your employer claim copyright, you must complete Part V and be sure to include a notice of copyright

with each such image in the paper.
(•) We/I do not have any artistic images.
( ) We/I have any artistic images.

VI. Representations, Warranties and Covenants

The undersigned hereby represents, warrants and covenants as follows:

(a) Owner is the sole owner or authorized agent of Owner(s) of the Work;

(b) The undersigned is authorized to enter into this Agreement and grant the rights included in this license to ACM;

(c) The Work is original and does not infringe the rights of any third party; all permissions for use of third-party materials consistent in scope and duration with the rights granted to ACM have been obtained, copies of such permissions have been provided to ACM, and the Work as submitted to ACM clearly and accurately indicates the credit to the proprietors of any such third-party materials (including any applicable copyright notice), or will be revised to indicate such credit;

(d) The Work has not been published except for informal postings on non-peer reviewed servers, and Owner covenants to use best efforts to place ACM DOI pointers on any such prior postings;

(e) The Auxiliary Materials, if any, contain no malicious code, virus, trojan horse or other software routines or hardware components designed to permit unauthorized access or to disable, erase or otherwise harm any computer systems or software; and

(f) The Artistic Images, if any, are clearly and accurately noted as such (including any applicable copyright notice) in the Submitted Version.

☑ I agree to the Representations, Warranties and Covenants

Funding Agents

1. National Science Foundation award number(s): 1618364,1350206

DATE: **03/08/2019** sent to qzzhao@mtu.edu at **14:03:15**

IEEE Requesting permission to reuse content from an IEEE publication

SAMG: Sparsified graph-theoretic algebraic multigrid for solving large symmetric diagonally dominant (SDD) matrices

Conference Proceedings:
2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)
Author: Zhiqiang Zhao; Yongyu Wang; Zhuo Feng
Publisher: IEEE
Date: 13-16 Nov. 2017

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

ACM Copyright and Audio/Video Release

Title of the Work: A Spectral Graph Sparsification Approach to Scalable Vectorless Power Grid Integrity Verification

Author/Presenter(s): Zhiqiang Zhao;Zhuo Feng
Type of material: Full Paper

Publication and/or Conference Name: DAC '17: The 54th Annual Design Automation Conference 2017 Proceedings

I. Copyright Transfer, Reserved Rights and Permitted Uses

* Your Copyright Transfer is conditional upon you agreeing to the terms set out below.

Copyright to the Work and to any supplemental files integral to the Work which are submitted with it for review and publication such as an extended proof, a PowerPoint outline, or appendices that may exceed a printed page limit, (including without limitation, the right to publish the Work in whole or in part in any and all forms of media, now or hereafter known) is hereby transferred to the ACM (for Government work, to the extent transferable) effective as of the date of this agreement, on the understanding that the Work has been accepted for publication by ACM.

Reserved Rights and Permitted Uses

(a) All rights and permissions the author has not granted to ACM are reserved to the Owner, including all other proprietary rights such as patent or trademark rights.

(b) Furthermore, notwithstanding the exclusive rights the Owner has granted to ACM, Owner shall have the right to do the following:

(i) Reuse any portion of the Work, without fee, in any future works written or edited by the Author, including books, lectures and presentations in any and all media.

(ii) Create a "Major Revision" which is wholly owned by the author

(iii) Post the Accepted Version of the Work on (1) the Author's home page, (2) the Owner's institutional repository, (3) any repository legally mandated by an agency funding the research on which the Work is based, and (4) any non-commercial repository or aggregation that does not duplicate ACM tables of contents, i.e., whose patterns of links do not substantially duplicate an ACM-copyrighted volume or issue. Non-commercial repositories are here understood as repositories owned by non-profit organizations that do not charge a fee for accessing deposited articles and that do not sell advertising or otherwise profit from serving articles.

(iv) Post an "Author-Izer" link enabling free downloads of the Version of Record in the ACM Digital Library on (1) the Author's home page or (2) the Owner's institutional repository;

(v) Prior to commencement of the ACM peer review process, post the version of the Work as submitted to ACM ("Submitted Version" or any earlier versions) to non-peer reviewed servers;

(vi) Make free distributions of the final published Version of Record internally to the Owner's employees, if applicable;

(vii) Make free distributions of the published Version of Record for Classroom and Personal Use;

Are any of the co-authors, employees or contractors of a National Government? ○ Yes ◉ No

---

II. PERMISSION FOR CONFERENCE TAPING AND DISTRIBUTION

Audio/Video Release
* Your Audio/Video Release is conditional upon you agreeing to the terms set out below.

I hereby grant permission for ACM to include my name, likeness, presentation and comments in any and all forms, for the Conference and/or Publication.

I further grant permission for ACM to record and/or transcribe and reproduce my presentation as part of the ACM Digital Library, and to distribute the same for sale in complete or partial form as part of an ACM product on CD-ROM, DVD, webcast, USB device, streaming video or any other media format now or hereafter known.

I understand that my presentation will not be sold separately as a stand-alone product without my direct consent. Accordingly, I give ACM the right to use my image, voice, pronouncements, likeness, and my name, and any biographical material submitted by me, in connection with the Conference and/or Publication, whether used in excerpts or in full, for distribution described above and for any associated advertising or exhibition.

Do you agree to the above Audio/Video Release? ◉ Yes ○ No

III. Auxiliary Material

Do you have any Auxiliary Materials? ○ Yes ◉ No

IV. Third Party Materials

In the event that any materials used in my presentation or Auxiliary Materials contain the work of third-party individuals or organizations (including copyrighted music or movie excerpts or anything not owned by me), I understand that it is my responsibility to secure any necessary permissions and/or licenses for print and/or digital publication, and cite or attach them below.

◉ We/I have not used third-party material.
○ We/I have used third-party materials and have necessary permissions.

V. Artistic Images

If your paper includes images that were created for any purpose other than this paper and to which you or your employer claim copyright, you must complete Part V and be sure to include a notice of copyright with each such image in the paper.
◉ We/I do not have any artistic images.
○ We/I have any artistic images.

VI. Representations, Warranties and Covenants

The undersigned hereby represents, warrants and covenants as follows:

(a) Owner is the sole owner or authorized agent of Owner(s) of the Work;

(b) The undersigned is authorized to enter into this Agreement and grant the rights included in this license to ACM;

(c) The Work is original and does not infringe the rights of any third party; all permissions for use of third-party materials consistent in scope and duration with the rights granted to ACM have been obtained, copies of such permissions have been provided to ACM, and the Work as submitted to ACM clearly and accurately indicates the credit to the proprietors of any such third-party materials (including any applicable copyright notice), or will be revised to indicate such credit;

(d) The Work has not been published except for informal postings on non-peer reviewed servers, and Owner covenants to use best efforts to place ACM DOI pointers on any such prior postings;

(e) The Auxiliary Materials, if any, contain no malicious code, virus, trojan horse or other software routines or hardware components designed to permit unauthorized access or to disable, erase or otherwise harm any computer systems or software; and

(f) The Artistic Images, if any, are clearly and accurately noted as such (including any applicable copyright notice) in the Submitted Version.

☑ I agree to the Representations, Warranties and Covenants

Funding Agents

1. Division of Computing and Communication Foundations award number(s): 1618364

DATE: **03/17/2017** sent to qzzhao@mtu.edu at **16:03:13**

IEEE Requesting permission to reuse content from an IEEE publication

A Spectral Approach to Scalable Vectorless Thermal Integrity Verification

Conference Proceedings:
2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)

Author: Zhiqiang Zhao; Zhuo Feng

Publisher: IEEE

Date: 9-13 March 2020

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW