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

Zhiqiang Zhao
Department of ECE
Michigan Technological University
Houghton, Michigan 49931
Email: qzzhao@mtu.edu

Yongyu Wang
Department of ECE
Michigan Technological University
Houghton, Michigan 49931
Email: yongyuw@mtu.edu

Zhuo Feng
Department of ECE
Michigan Technological University
Houghton, Michigan 49931
Email: zhuofeng@mtu.edu

***Abstract*—Algebraic multigrid (AMG) is a class of high-performance linear solvers 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 according to the input matrices. Graph-theoretic Algebraic Multigrid (AMG) algorithms have emerged for solving large Symmetric Diagonally Dominant (SDD) matrices by taking advantages of spectral properties of graph Laplacians. This paper proposes a Sparsified graph-theoretic Algebraic Multigrid (SAMG) framework that allows efficiently constructing 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.**

***Keywords*—Algebraic multigrid, graph Laplacian matrix, spectral graph theory, graph sparsification.**

I. Introduction

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 [1]–[4], spectral graph partitioning and data clustering problems [4], semi-supervised learning (SSL) [5], as well as interior-point problems for maximum flow of undirected graphs [6].

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 [7] and the Lean Algebraic Multigrid (LAMG) solver [8]. One common feature of these multigrid algorithms is that they construct the coarse level problems by taking advantages 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 [7], whereas the LAMG solver applies node elimination and node aggregation to form the coarse level problems [8]. Although such graph-theoretic AMG algorithms can significantly improve the efficiency and scalability for solving large SDD matrix problems than 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 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, 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 [9], 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 improves the scalability of existing graph-theoretic AMG methods that are already very efficient.

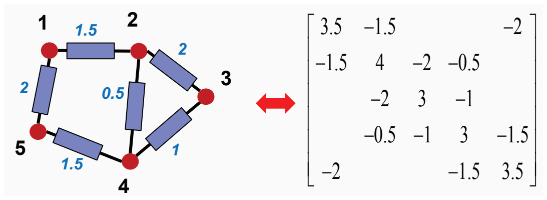
The rest of this paper is organized as follows: the background of graph Laplacian matrix and spectral graph sparsification, as well as state-of-the-art graph-theoretic AMG methods, has been provided in Section II. In Section III, we introduce an efficient Sparsified Algebraic Multigrid (SAMG) method for solving large SDD matrices. Section IV demonstrates extensive experimental results of the SAMG solver for a variety of real-world, large-scale sparse SDD matrices to validate the proposed approach, which is followed by the conclusion of this work in Section V.

II. Background

A. Spectral Sparsification of Graph Laplacian Matrices

As shown in Fig. 1, the Laplacian matrix of graph $G = (V, E, w)$ can be defined as follows, where V is a set of vertices, E is a set of edges, and w is a weight function that assigns a positive weight to every edge:

$$L_G(p,q) = \begin{cases} -w(p,q) & \text{if } (p,q) \in E \\ \sum\limits_{(p,t)\in E} w(p,t) & \text{if } (p=q) \\ 0 & \text{if } otherwise. \end{cases} \quad (1)$$



$$\begin{bmatrix} 3.5 & -1.5 & & & -2 \\ -1.5 & 4 & -2 & -0.5 & \\ & -2 & 3 & -1 & \\ & -0.5 & -1 & 3 & -1.5 \\ -2 & & & -1.5 & 3.5 \end{bmatrix}$$

Fig. 1. A resistor network (conductance value of each element is shown) and its graph Laplacian matrix.

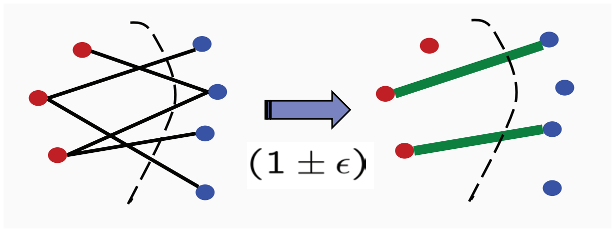


Fig. 2. Cut sparsifier preserves cuts of a graph.

It can be shown that the graph Laplacian matrix is a SDD matrix, which can be considered as an admittance matrix of a resistive circuit network [1]. For any real vector $x \in \mathbb{R}^V$, the Laplacian quadratic form of graph G can be defined as follows:

$$x^T L_G x = \sum_{(p,q)\in E} w_{p,q}(x(p) - x(q))^2. \quad (2)$$

Graph sparsification aims to find an ultra sparse subgraph (a.k.a graph sparsifier) such that the subgraph can preserve all vertices but significantly less number of edges than the original graph. There are two types of graph sparsification methods: the cut-based graph sparsification method [10] and spectral graph sparsification method [3]. The cut sparsifier preserves the values of cuts in a graph as shown in Fig. 2, whereas spectral sparsifier preserves Laplacian quadratic forms as well as eigenvalues/eigenvectors of the original graph. It has been shown that spectral sparsifier is always a stronger notion than the cut sparsifier [3].

Two graphs are spectrally similar if they have similar quadratic forms as shown in Fig. 3. More specifically, graph G and P are said to be $\sigma-$spectrally similar if for all real vectors $x \in \mathbb{R}^V$ the following holds [11]:

$$\frac{x^T L_P x}{\sigma} \le x^T L_G x \le \sigma x^T L_P x. \quad (3)$$

The relative condition number can be computed by:

$$\kappa(L_G, L_P) = \lambda_{max}/\lambda_{min} \le \sigma^2, \quad (4)$$

where λ_{max} (λ_{min}) is the largest (smallest non-zero) eigenvalue of matrix $L_P^+ L_G$ with L_P^+ denoting the Moore-Penrose pseudoinverse of L_P matrix. It has been shown that a spectrally similar subgraph can approximately preserve the distances or effective resistances between vertices [12] so that much faster graph-based algorithms can be developed based on these "spectrally" sparsified networks. It also should be noted that a graph sparsifier with a smaller relative condition number (higher spectral similarity) can lead to faster convergence when used as preconditioners in iterative methods, such as Krylov-subspace iterative methods.

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 $n = |V|$ [13]. 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 [1], [2], [4], [9], [12], [14], 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 Fig. 4. 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 [9], [12]. 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.

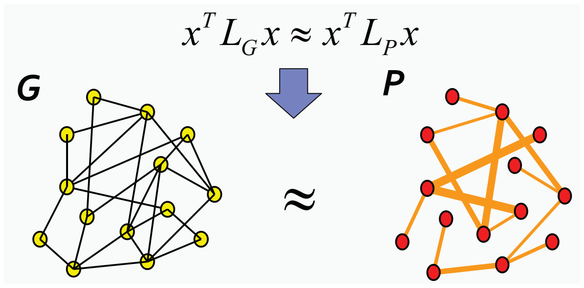


Fig. 3. Two spectrally similar graphs G and P.

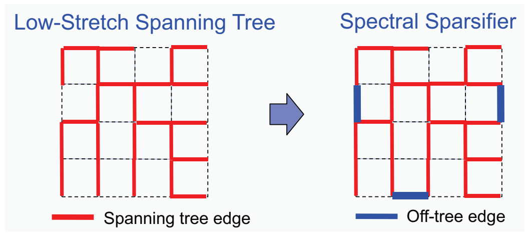


Fig. 4. A spanning tree and its ultra-sparsifier subgraph.

B. Graph-theoretic Algebraic Multigrid (AMG)

Algebraic Multigrid (AMG) [15] solvers have been developed for solving large sparse matrices based on the 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 not only be fast and scalable, but also reliable and robust for different kinds of input matrices. Recent Combinatorial Multigrid solver (CMG) [7] and Lean Algebraic Multigrid solver (LAMG) [8] are the 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 that is similar to the construction of a quotient graph [7]. However, the coarse level problems (graphs) obtained by CMG will usually become increasingly denser, 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 will grow very fast, which can significantly impact

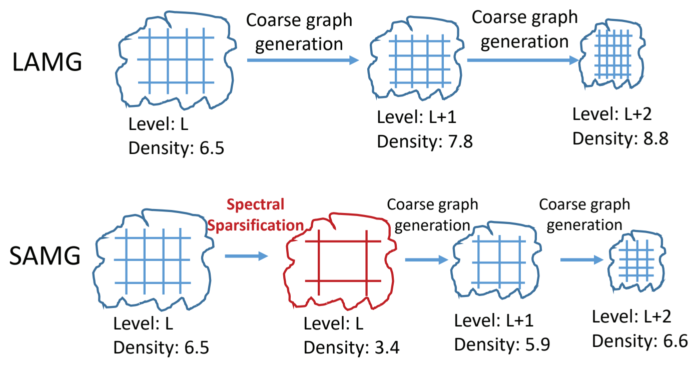


Fig. 5. Comparison of the setup phases between LAMG [8] and SAMG (this work).

the CMG solver speed as well efficiency for parallel computing (due to the high communication cost).

Another well known graph-theoretic AMG solver is the LAMG solver [8] 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 wider range of applications [8]. However, the LAMG solver may run into the 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.

III. Sparsified Algebraic Multigrid

A. *Overview of Our Method*

The proposed SAMG solver in this work is built upon the framework of the prior LAMG solver [8]. During the SAMG setup phase, we introduce a graph sparsification procedure based on a recent spectral perturbation based spectral graph sparsification approach [9] 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 has been shown in Fig.5, while detailed steps for setting up a coarser level $(l+1)$ problem from an existing coarse level (l) problem in SAMG have been briefly described as follows:

(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 [8].

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

B. *Spectral Graph Sparsification via Spectral Perturbation Analysis*

In this section, we introduce a practically efficient, nearly-linear time spectral graph sparsification algorithm [9] that will be integrated into the proposed SAMG solver for maintaining desired graph densities of coarse level problems. In the following sections, assume that a weighted, undirected and connected graph $G=(V,E,w)$ has a subgraph $P=(V_s,E_s,w_s)$ (such as a spanning-tree subgraph) with descending generalized eigenvalues $\lambda_{max}=\lambda_1\geq\lambda_2\geq\cdots\geq\lambda_n=0$, while their Laplacian matrices satisfy (5) for $i=1,...,n$, respectively:

$$L_G u_i = \lambda_i L_P u_i. \tag{5}$$

Recent research work proves that when a spanning tree subgraph is used as an initial spectral graph sparsifier, it has at most k eigenvalues greater than $\frac{\mathrm{st}_P(G)}{k}$ [16], where $\mathrm{st}_P(G)$ denotes the stretch of the original graph G with respect to the spanning tree subgraph P that is an upper bound of the largest generalized eigenvalue λ_{max}, and thus an upper bound for the condition number of $L_P^+L_G$ [16].

It also has been shown that if each of the largest eigenvalues can be fixed by adding a small number of extra off-tree edges to the spanning tree, an ultra-sparsifier with totally $n+o(n)$ edges can be created to provide a good spectral approximation of G [1], [2], [4], [14]. To this end, a linear-time spectral perturbation based algorithm has been recently proposed for identifying such "spectrally critical" edges [9], which is described as follows. Consider the following first-order eigenvalue perturbation problem:

$$L_G\left(u_i+\delta u_i\right)=\left(\lambda_i+\delta\lambda_i\right)\left(L_P+\delta L_P\right)\left(u_i+\delta u_i\right), \tag{6}$$

where a perturbation δL_P (Laplacian of off-tree edges) is applied to L_P (Laplacian of the spanning tree), leading to perturbations in generalized eigenvalues and eigenvectors $\lambda_i+\delta\lambda_i$ and $u_i+\delta u_i$ for $i=1,...,n-1$, respectively. The key to effective spectral sparsification is to identify distinct off-tree edges that will result in the greatest reduction of the dominant generalized eigenvalues. To this end, the following spectral off-tree edge embedding scheme has been proposed which starts with an initial random vector [9]:

$$x_0=\sum_{i=1}^{n}\alpha_i u_i, \tag{7}$$

where u_i for $i=1,...,n-1$ are L_P-orthogonal generalized eigenvectors that satisfy:

$$u_i^T L_P u_j=\begin{cases}1, i=j\\ 0, i\neq j.\end{cases} \tag{8}$$

Performing t-step power iterations to the generalized eigenvalue problem (5) leads to:

$$x_t=\left(L_P^+L_G\right)^t x_0=\sum_{i=1}^{n-1}\alpha_i\lambda_i^t u_i. \tag{9}$$

If we further define:

$$\delta L_{P,max}=L_G-L_P, \tag{10}$$

which can be considered as an extreme-case Laplacian matrix that includes all off-tree edges that belong to the original graph G but not

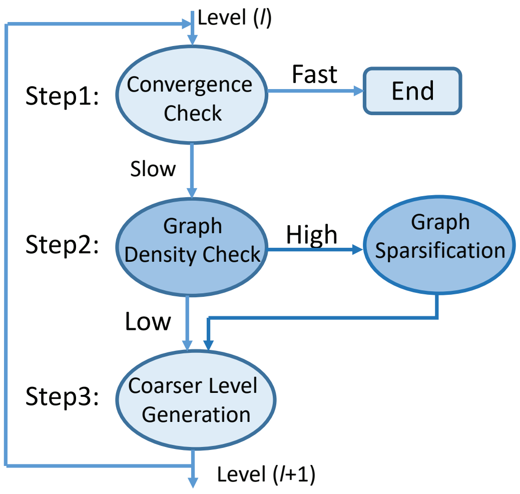


Fig. 6. Flowchart for the SAMG solver setup phase.

the subgraph P, then the following holds:

$$Q_{\delta L_{P,max}}(x_t) = {x_t}^T \delta L_{P,max} x_t = \sum_{i=1}^{n-1} \left(\alpha_i \lambda_i^t\right)^2 (\lambda_i - 1)$$
$$= \sum_{(p,q)\in E\setminus E_s} w_{p,q} \sum_{i=1}^{n-1} \alpha_i^2 \lambda_i^{2t} \left({u_i}^T e_{p,q}\right)^2 . \tag{11}$$

Interestingly, (11) will allow ranking each off-tree edge according to its "spectral-criticality" level, which also indicates that adding some of these off-tree edges with large Joule heat values back to P will immediately reduce the dominant generalized eigenvalues, thereby forming a high-quality spectral graph sparsifier [9]. It has been shown in [9] that a σ-similar spectral sparsifier with $n-1+O(\frac{m \log n \log \log n}{\sigma^2})$ edges can be obtained in $O(m)$ time using the spectral perturbation based method when an initial low-stretch spanning tree is given [17]. In the rest of this paper, the above spectral sparsification method will be effectively leveraged in the proposed SAMG solver when a high graph density is encountered on a coarse level problem.

C. Sparsified Algebraic Multigrid (SAMG)

The complete SAMG setup flow is depicted in Fig. 6, 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 [8], 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.

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 well control coarse level graph densities for all hierarchical levels, it would be more effective to sparsify finer level problems with larger sizes than sparsifying coarser level problems with smaller sizes; in another word, 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 (4).

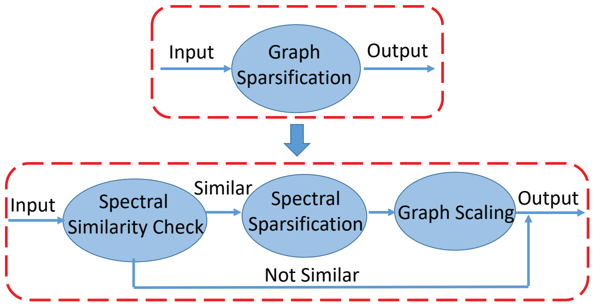


Fig. 7. Graph sparsification during the SAMG solver setup phase.

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:

$$\omega_l = \frac{nnz_l}{nnz_{l-1}}, \tag{12}$$

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:

$$\theta_l = \frac{agd_l}{agd_{l-1}}. \tag{13}$$

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 are 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.

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 sparsification and graph Laplacian scaling steps as illustrated in Fig. 7.

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

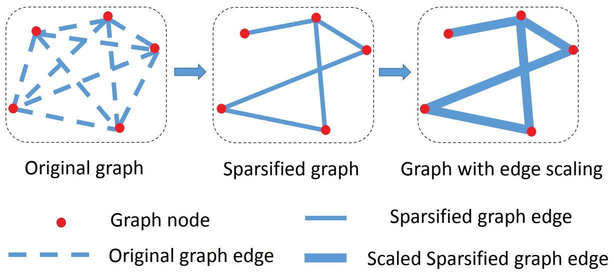


Fig. 8. Spectral graph sparsification with graph scaling.

as follows:

$$\Delta_l = \prod_{s<l} \kappa(L_{G_s}, L_{P_s}), \tag{14}$$

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 Fig.8 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 [18]:

$$\alpha_l = \frac{\sum\limits_{(p,q)\in G_l} w_{p,q}}{\sum\limits_{(p,q)\in P_l} \tilde{w}_{p,q}}, \tag{15}$$

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) 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 [8]:

$$c_{uv} = \frac{\|(X_u, X_v)\|^2}{(X_u, X_u)(X_v, X_v)}, (X_u, X_v) = \Sigma_{k=1}^{K}(x_u^{(k)} \cdot x_v^{(k)}) \tag{16}$$

where $X = (x^{(1)} \dots x^{(K)})$ 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 [8]. Consequently, nodes with large affinity values can be aggregated together to form the nodes on the coarser level graph.

IV. 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 sparse matrix collection of

TABLE I. 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

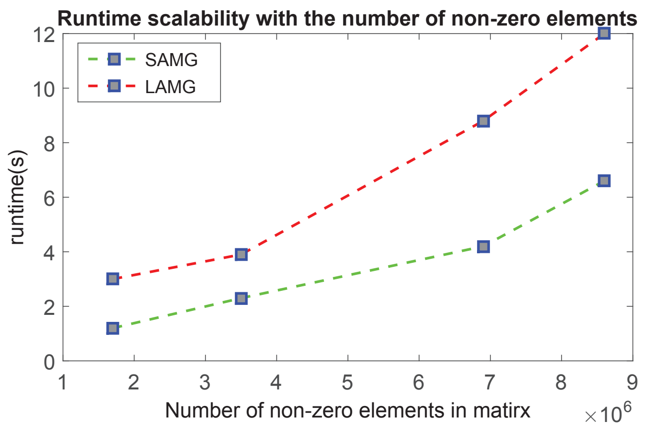


Fig. 9. Runtime scalability with increasing number of nonzero elements.

The University of Florida [19], including matrices in 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 neighbours (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 RHEW 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 [8], 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 I. The systems $Ax = b$ 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 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.

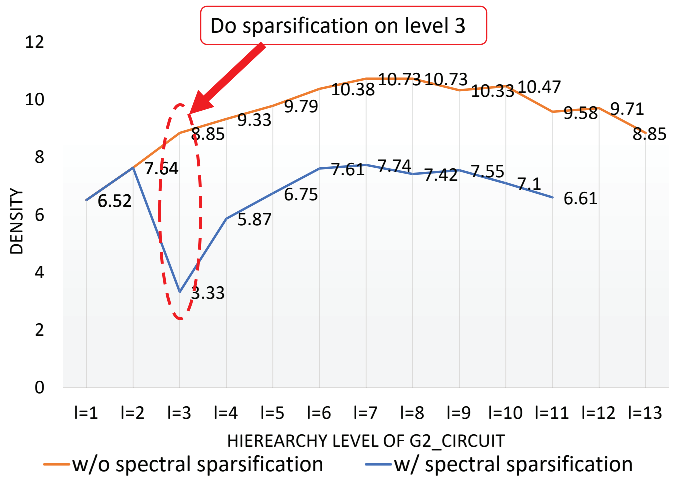


Fig. 10. Comparison of average graph densities of coarse level problems for G2_circuit matrix.

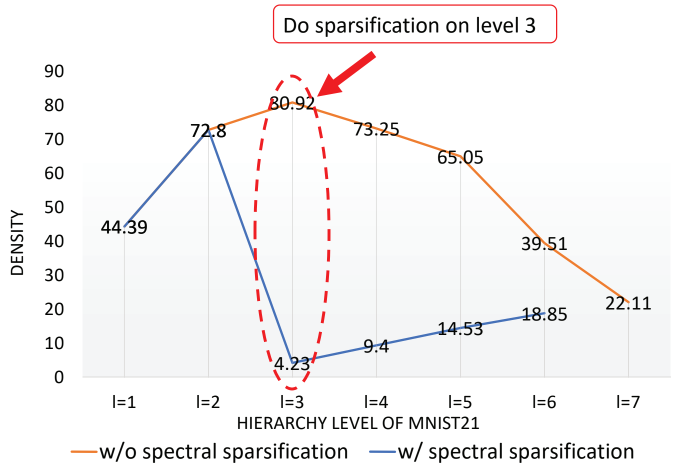


Fig. 11. Comparison of average graph densities of coarse level problems for MNIST21.

From Table I, 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. Fig. 9 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. Fig. 10, and Fig. 11 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 sparsified coarse level problems in SAMG are much lower than the ones in LAMG.

V. Conclusion

This paper aims to improve the scalability of recent graph-theoretic Algebraic Multigrid (AMG) solvers by introducing a Sparsified Algebraic Multigrid (SAMG) framework. 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, finite element analysis, as well as data mining and machine learning tasks.

VI. Acknowledgments

This work is supported in part by the National Science Foundation under Grants CCF-1350206 (CAREER), CCF-1618364 (SHF) and CCF-1318694 (SHF).

References

[1] D. Spielman, "Algorithms, graph theory, and linear equations in laplacian matrices," *Proceedings of the International Congress of Mathematicians Hyderabad*, 2010.

[2] I. Koutis, G. Miller, and R. Peng, "Approaching Optimality for Solving SDD Linear Systems," in *Proc. IEEE FOCS*, 2010, pp. 235–244.

[3] D. Spielman and S. Teng, "Spectral sparsification of graphs," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 981–1025, 2011.

[4] 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*, vol. 35, no. 3, pp. 835–885, 2014.

[5] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proc. of ICML*, 2003, vol. 3, pp. 912–919.

[6] 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*, 2011, pp. 273–282.

[7] 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*, vol. 115, no. 12, pp. 1638–1646, 2011.

[8] O. Livne and A. Brandt, "Lean algebraic multigrid (lamg): Fast graph laplacian linear solver," *SIAM Journal on Scientific Computing*, vol. 34, no. 4, pp. B499–B522, 2012.

[9] Z. Feng, "Spectral Graph Sparsification in Nearly-Linear Time Leveraging Efficient Spectral Perturbation Analysis," in *Proc. IEEE/ACM DAC*, 2016.

[10] A. Benczúr and D. Karger, "Approximating st minimum cuts in õ (n2) time," in *Proc. ACM STOC*, 1996, pp. 47–55.

[11] J. Batson, D. Spielman, N. Srivastava, and S. Teng, "Spectral sparsification of graphs: theory and algorithms," *Communications of the ACM*, vol. 56, no. 8, pp. 87–94, 2013.

[12] D. Spielman and N. Srivastava, "Graph sparsification by effective resistances," in *Proc. ACM STOC*, 2008, pp. 563–568.

[13] J. Batson, D. Spielman, and N. Srivastava, "Twice-ramanujan sparsifiers," *SIAM Journal on Computing*, vol. 41, no. 6, pp. 1704–1721, 2012.

[14] A. Kolla, Y. Makarychev, A. Saberi, and S. Teng, "Subgraph sparsification and nearly optimal ultrasparsifiers," in *Proc. ACM STOC*, 2010, pp. 57–66.

[15] J. Ruge and K. Stüben, "Algebraic multigrid," *Multigrid methods*, vol. 3, no. 13, pp. 73–130, 1987.

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

[17] I. Abraham and O. Neiman, "Using petal-decompositions to build a low stretch spanning tree," in *Proc. ACM STOC*, 2012, pp. 395–406.

[18] X. Zhao, Z. Feng, and C. Zhuo, "An efficient spectral graph sparsification approach to scalable reduction of large flip-chip power grids," in *Proc. of IEEE/ACM ICCAD*. IEEE Press, 2014, pp. 218–223.

[19] T. Davis and Y. Hu, "The University of Florida sparse matrix collection," *ACM Trans. on Math. Soft. (TOMS)*, vol. 38, no. 1, pp. 1, 2011.