I. Background

(1) Graph Sparsification Basics
- Goal: find a sparse subgraph (graph sparsifier) to approximate the original graph [1]
- The sparsifier should have the same set of vertices but much fewer edges
- Becomes key to designing nearly-linear time numerical & graph algorithms [12]
- Applications: solving PDEs and sparse matrices, graph partitioning (data clustering), semi-supervised learning (SSL), maximum flow of undirected graphs, etc

The original graph

The sparsified graph

Figure from [2]


(2) Laplacian of A Resistor Network

A Weighted Graph $A = (V, E, w)$

Symmetric Diagonally Dominant (SDD) Laplacian Matrix $G$

$$G(x, y) = \sum_{m \in \mathcal{E}} w(m) (x(m) - x(y(m)))^2$$

Quadratic Laplacian Form (Joule Heat)

$$x^T G x = \sum_{m \in \mathcal{E}} w(m) (x(m) - x(y(m)))^2$$

(3) Two Types of Graph Sparsifiers
- Graph sparsifiers for preserving graph cuts or graph Laplacian spectrums
- Cut sparsifiers: preserve cuts between vertices [Benczúr & Karger. STOC'96]
- Spectral sparsifiers: preserve eigenvalues & eigenvectors of graph Laplacians [Teng & Spielman, SIAM J. on Comp.'11]

Graphs G and P are $\sigma$-spectrally similar if their Laplacian quadratic forms satisfy:

$$x^T L_p x \leq x^T L_G x \leq \sigma x^T L_p x$$

Graph Sparsification

The original graph

The sparsified graph

Figure from [2]

II. Technical Challenges

(1) Graph Sparsifier for Solving SDD Matrices
- Preconditioned Conjugate Gradient (CG) is widely used for solving $P^{-1} C x = b$
- PCG needs $k \mathcal{O}(\sqrt{n}/\epsilon)$ iterations, where the condition number:

$$\kappa(P) \leq \sum_{i=1}^n \lambda_i$$

Graph sparsifier should result in fast convergence when used as a preconditioner in CG

Spectral Graph Sparsification in Nearly-Linear Time Leveraging (SDD) Laplacian Matrix

$$G(P) \approx \mathcal{O}(\sqrt{n}/\epsilon)$$

(2) Spanning Tree as a Graph Sparsifier

As shown below, tree preconditioners have very well separated large eigenvalues [2]

$$\lambda_1 \approx \lambda_2 \approx \cdots \approx \lambda_k$$

It is possible to dramatically reduce the largest eigenvalues by adding some off-tree edges

(3) Fixing A Group of Eigenvalues
- Perform 1-step generalized power iteration with a random vector $x_0$
- Compute the Joule heat of each off-tree edge with the eigenvector $\delta \lambda_i = \left| \lambda_i - \lambda \right|$

$$\lambda_i \Rightarrow \lambda_{\text{fixed}} \Rightarrow \lambda_i$$

Sequence for fixing largest eigenvalues

Challenge: There can be too many large eigenvalues!

(4) Nearly-Linear Time Complexity
- Low-stretch spanning trees can be extracted in nearly linear time [2]
- Generalized power iteration for a spanning tree can be done in linear time

IV. Experiment Results

(1) Spectral Graph Sparsification Results
- 2D mesh grid ($n=40000$) with unit edge weight
- 400 off-tree edges added to the “hair comb” spanning tree
- $\lambda_{\text{max}}$ is reduced from 6E4 to 1E2: 640X reduction!

2D mesh

Adding off-tree edges

Initial Joule Heat Map

Final Joule Heat Map

O(n<sup>2</sup>) vs. n<sup>1.5</sup>

(2) SDD Solver for Power Grid Analysis

- Trillion scale power grid analysis
- $G$ is a $n \times n$ matrix with $O(n^2)$ entries
- $G$ is sparse with $O(n)$ non-zeros

Test Cases

<table>
<thead>
<tr>
<th>Case</th>
<th># of Nodes</th>
<th># of Non-Zeros</th>
<th>Direct Method (Cholesky)</th>
<th>Ultra-Solver (Cholmod)</th>
<th>Ultra-Solver (MUMPS)</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>G3_circuit</td>
<td>1.6E6</td>
<td>7.7E6</td>
<td>45s (2.32)</td>
<td>37 iter. 9%</td>
<td>5.2s (21594)</td>
<td>45,897X</td>
</tr>
<tr>
<td>thermal2</td>
<td>1.2E6</td>
<td>8.6E6</td>
<td>16.5s (1.60)</td>
<td>34 iter. 10%</td>
<td>4.4s (22756)</td>
<td>1,952X</td>
</tr>
<tr>
<td>ecology2</td>
<td>1.0E6</td>
<td>5.8E6</td>
<td>12.5s (0.70)</td>
<td>47 iter. 8%</td>
<td>3.4s (24386)</td>
<td>1,728X</td>
</tr>
<tr>
<td>tmt_sym</td>
<td>0.7E6</td>
<td>5.1E6</td>
<td>11.5s (0.99)</td>
<td>30 iter. 10%</td>
<td>2.2s (21686)</td>
<td>796X</td>
</tr>
<tr>
<td>parabolion</td>
<td>0.5E6</td>
<td>3.7E6</td>
<td>6.3s (401M)</td>
<td>25 iter. 9%</td>
<td>1.2s (979M)</td>
<td>120X</td>
</tr>
</tbody>
</table>

V. Conclusion

- Spanning trees are critical for building high-quality spectral graph sparsifiers
- Spectral critical off-tree edges can be efficiently identified by the proposed spectral perturbation approach
- Nearly linear time algorithms for graph sparsification and solving SDD matrices have been developed showing good progress

VI. Acknowledgements

- NSF CCF CAREER Grant #1502006
- NSF CCF SHF Grant #1318694
- MTU Research Enhancement Fund
- PhD students: Xuejuan Zhao, Linglei Han