Towards Practically-Efficient Spectral Sparsification of Graphs

Zhuo Feng

Acknowledgements:
PhD students: Xueqian Zhao, Lengfei Han, Zhiqiang Zhao, Yongyu Wang
NSF CCF CAREER grant #1350206, NSF CCF grants #1318694 & #1618364
Increasing Complexity of Graphs (Networks)

- A typical graph or network: nodes + (weighted) edges

<table>
<thead>
<tr>
<th></th>
<th>Trans. Network</th>
<th>Social Network</th>
<th>Circuit Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>cities</td>
<td>people</td>
<td>points of electrical connections</td>
</tr>
<tr>
<td>Edges</td>
<td>roads</td>
<td>friendships</td>
<td>resistors, capacitors, inductors, etc</td>
</tr>
</tbody>
</table>

Many graph (network) problems are only getting bigger!
- Data and social networks are evolving (growing) every second
- Integrated circuit networks are doubling in size every other year

How to deal with fast growing massive graphs and networks?
What is Graph Sparsification?

- Find a graph proxy (sparsifier) to mimic the original graph
  - w/ the same set of nodes but much fewer edges

![The original graph](image1)  ![The sparsified graph](image2)

- Key to designing fast numerical & graph algorithms:
  - Solving partial differential equations (PDEs) and sparse matrices, graph partitioning, data mining, graph-based semi-supervised learning (SSL), maximum flow of undirected graphs, big data graph analytics...
  - Also known as the Laplacian paradigm

- Key enabler for efficient handling of big (data) graphs
  - Big graph computing on personal computers, or even mobile devices
Cut and Spectral Graph Sparsifiers

- **Cut sparsifiers** preserve cuts between nodes \( (\text{Benczúr & Karger. STOC'96}) \)
  - Randomly sample each edge with a probability
  - Adjust the edge weight if included in the sparsifier

- **Spectral sparsifiers** preserve more: \( (\text{Spielman & Teng. SIAM J. Comp.'11}) \)
  - Eigenvalues & eigenvectors of Laplacian matrices
  - Pair-wise distances, effective resistance, cuts between nodes, …
  - A much stronger notion than cut sparsifier
Graph Laplacian Matrix (Admittance Matrix)

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

Graph Laplacian $L_G$

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

Graph Laplacian is a Symmetric, Diagonally Dominant (SDD) Matrix

$L_G(u,v) = \begin{cases} 
-w(u,v) & \text{if } (u,v) \in E \\
\sum_{(u,v) \in E} w(u,v) & \text{if } u = v \\
0 & \text{otherwise}
\end{cases}$
Laplacian Quadratic Form

Laplacian Quadratic form of a graph $G = (V, E, w)$:

$$x^T L_G x = \sum_{(u,v) \in E} w(u, v) (x(u) - x(v))^2$$

Let's look at a resistor network!

- $x$ vector $\Rightarrow$ voltage vector
- edge weight $\Rightarrow$ conductance
- quadratic form $\Rightarrow$ total Joule heat

$$x^T L_G x = 1.5(0.5 - 0)^2 + 2(1.5 - 0.5)^2$$
$$+1(4 - 1.5)^2 + 0.5(4 - 0.5)^2$$
$$+1.5(4 - 2.5)^2 + 2(2.5 - 0)^2$$
$$= 30.625 \ (Watt)$$
Spectral Graph Similarity

If for all real vectors $x$:

$$x^T L_G x \approx x^T L_P x$$

Graphs $G$ and $P$ are $\sigma$-spectrally similar if for all real $x$:

$$\frac{x^T L_P x}{\sigma} \leq x^T L_G x \leq \sigma x^T L_P x$$
Iterative Matrix Solvers

Total cost for Conjugate Gradient (CG) to solve $Ax = b$

\[ \sqrt{\kappa(A)}m \log \left( \frac{1}{\varepsilon} \right) \]

Condition Number:

\[ \kappa(A) = \frac{\lambda_{\text{max}}(A)}{\lambda_{\text{min}}(A)} \]

Cost for mat-vec multiply

error tolerance
Preconditioned Conjugate Gradient (PCG)

Find $B$ that can well approximate $A$
Solve $B^{-1}Ax = B^{-1}b$ instead!

\[ \sqrt{\kappa(B^{-1}A)} \left( m + \text{cost for solving } B \right) \log \left( \frac{1}{\varepsilon} \right) \]

Relative Condition Number:

\[ \kappa(B^{-1}A) = \frac{\lambda_{\text{max}}(B^{-1}A)}{\lambda_{\text{min}}(B^{-1}A)} \]

preconditioning time
Spectral Sparsifiers as Preconditioners

- **PCG with graph sparsifiers as preconditioners**
  - Solve $L_P^+L_Gx = L_P^+b$ instead
  - Smaller relative condition number means faster convergence

Relative Condition Number:

$$\kappa(L_P^+L_G) = \frac{\lambda_{\max}(L_P^+L_G)}{\lambda_{\min}(L_P^+L_G)} = \sigma^2$$

Figure: Simulating Pressure over Airfoil. Source: http://www.nada.kth.se
Prior Art: Spectral Sparsification by Effective Resistance

- **Sample edges by effective resistances** [1]

  **Edge Effective Resistance:**
  
  \[ R_{p,q}^{\text{eff}} = (e_p - e_q)^T L_G^+ (e_p - e_q) \]

  where: \( e_{k} = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}^T \)

  **Challenge:** [1] needs to solve the original matrix!

- **Edge Sampling Probability:**

  \( p_e \propto w_{p,q} R_{p,q}^{\text{eff}} = \frac{R_{p,q}^{\text{eff}}}{R_{p,q}} \)

  [2] uses stretch values for sampling off-tree edges

---


Spanning-Tree Preconditioner

- **Spanning trees are the simplest graph sparsifiers**
  - Recent research focuses on low-stretch spanning trees (LSSTs) [1]
  - Allows to solve SDD matrices in nearly-linear time [2]

---

**Original Matrix**

$$
\begin{bmatrix}
4 & -2 & -1 & 0 & 0 & 0 & 0 & 0 \\
-2 & 9 & -4 & 0 & -3 & 0 & 0 & 0 \\
0 & -4 & 12 & 0 & 0 & -8 & 0 & 0 \\
-1 & 0 & 11 & -6 & 0 & -4 & 0 & 0 \\
0 & -3 & 0 & -6 & 15 & -5 & 0 & -1 \\
0 & 0 & -8 & 0 & -5 & 16 & 0 & 0 & -3 \\
0 & 0 & 0 & -4 & 0 & 0 & 13 & -9 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & -9 & 14 & -4 \\
0 & 0 & 0 & 0 & 0 & -3 & 0 & -4 & 8
\end{bmatrix}
$$

**Spar. Mat**

$$
\begin{bmatrix}
3 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-2 & 6 & -4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -4 & 12 & 0 & 0 & -8 & 0 & 0 & 0 \\
0 & 0 & 0 & 10 & -6 & 0 & -4 & 0 & 0 \\
0 & 0 & 0 & -6 & 11 & -5 & 0 & 0 & 0 \\
0 & 0 & -8 & 0 & -5 & 13 & 0 & 0 & 0 \\
0 & 0 & 0 & -4 & 0 & 0 & 13 & -9 & 0 \\
0 & 0 & 0 & 0 & 0 & -9 & 13 & -4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -4 & 8 & 5
\end{bmatrix}
$$

<table>
<thead>
<tr>
<th>Mat./Eig.</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
<th>6th</th>
<th>Cond.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spar. Mat</td>
<td>23.442</td>
<td>22.271</td>
<td>15.877</td>
<td>9.095</td>
<td>7.0266</td>
<td>4.557</td>
<td>139.643</td>
</tr>
<tr>
<td>$\kappa(L_p^+L_G)$</td>
<td>3.459</td>
<td>2.881</td>
<td>1.431</td>
<td>1.180</td>
<td>1.000</td>
<td>1.000</td>
<td>3.459</td>
</tr>
</tbody>
</table>

Low-Stretch Spanning Tree (LSST)

- Is spanning tree always a good approximation?

The stretch of an edge \((p,q)\):

\[
	ext{st}_P(p,q) = \frac{\text{dist}_P(p,q)}{\text{dist}_G(p,q)} = w_{p,q} \left( \sum_{f \in P} \frac{1}{w_f} \right)
\]

The total stretch of graph \(G\):

\[
	ext{st}_P(G) = \sum_{(p,q) \in E} \text{st}_P(p,q)
\]

Every graph has an LSST such that [1]:

\[
	ext{st}_P(G) \leq O\left(m \log n \log \log \log n\right)
\]

\(m\): # of edges, \(n\): # of nodes

Spectral Similarity of A Tree Sparsifier

The total stretch is equal to the trace [1]:

\[ \text{st}_P(G) = \text{Tr}(L_P^+L_G) \]

Proof:

\[ \text{Tr}(L_P^+L_G) = \sum_{i=1}^{n} \lambda_i \]

\[ = \sum_{(p,q)\in E} w_{p,q} \text{Tr}(L_P^+(e_p - e_q)(e_p - e_q)^T) \]

\[ = \sum_{(p,q)\in E} w_{p,q} (e_p - e_q)^T L_P^+(e_p - e_q) \]

\[ = \text{st}_P(G) > \lambda_{\max} \]

The relative condition number is bounded by the total stretch [1]:

\[ 1 \leq \kappa(L_P^+L_G) = \sigma^2 \leq \lambda_{\max} \leq \text{st}_P(G) \leq O(m \log n \log \log n) \]

Generalized Eigenvalues of Tree Preconditioners

- Well separated large generalized eigenvalues \([1]\)

\[
\lambda_k \ldots \lambda_3 \lambda_2 \lambda_1
\]

\[
\frac{\text{st}_P(G)}{k} \frac{\text{st}_P(G)}{3} \frac{\text{st}_P(G)}{2} \frac{\text{st}_P(G)}{1}
\]

- Dramatically reduce \(\kappa(L^+_P L_G)\) by adding a few off-tree edges

Our Result: A Spectral Perturbation Framework

- First-order eigenvalue perturbation analysis \[1\]

\[
L_G \left( u_i + \delta u_i \right) = \left( \lambda_i + \delta \lambda_i \right) \left( L_P + \delta L_P \right) \left( u_i + \delta u_i \right)
\]

\[
\Rightarrow L_G \delta u_i = \lambda_i L_P \delta u_i + \delta \lambda_i L_P u_i + \lambda_i \delta L_P u_i
\]

\[
\delta L_P = \sum_{(p,q) \in G \setminus P} w_{p,q} \left( e_p - e_q \right) \left( e_p - e_q \right)^T
\]

\[1\] Z. Feng. Spectral Graph Sparsification in Nearly-Linear Time Leveraging Efficient Spectral Perturbation Analysis. ACM/IEEE DAC’16
Spectral Perturbation and Graph Embedding

- Eigenvalue perturbation proportional to edge Joule heat

\[ \delta \lambda_i \propto u_i^T \delta L_P u_i = \sum_{(p,q) \in G\setminus P} w_{p,q} u_i^T (e_p - e_q)(e_p - e_q)^T u_i \]

Embed edges using generalized eigenvectors [1]

\[ V_p = u_i(p) \quad V_q = u_i(q) \]

Graph Laplacian Eigenvectors

- **Laplacian quadratic form measures the boundary size**

  \[ x(u) = \begin{cases} 
  1 & \text{if node } u \text{ is in } S \\
  0 & \text{otherwise} 
  \end{cases} \]

  Edges going out \( S \) (boundary): \( x^T L_G x = \text{boundary}(S) \)

  **Courant – Fischer theorem:**

  \[
  \tau_{\text{max}} = \max_{\|x\|=1} x^T L_G x \Rightarrow \max \text{boundary}(S) \\
  \tau_{\text{min}} = \min_{\|x\|=1} x^T L_G x \Rightarrow \min \text{boundary}(S)
  \]

- **Embed nodes using eigenvectors**

  - Approximate max/min cuts in a graph
  - Key ideas behind spectral partitioning/clustering [1]

---

Why Generalized Eigenvectors?

Courant – Fischer theorem for generalized problems

\[
\lambda_{\text{max}} = \max_{\|x\|=1} \frac{x^T L_G x}{x^T L_P x} \Rightarrow \frac{\max \text{ cuts in } G}{\min \text{ cuts in } P} \Rightarrow \max \text{ mismatch in } P
\]
Incremental Spectral Sparsification (A Naïve Approach)

1. Extract an LSST;
2. Repeat the following:
   1) Find the \textbf{dominant eigenvector}
   2) Rank off-tree edges by \textbf{Joule heat}
   3) Add top off-tree edges to the tree

\textbf{Challenge:}
How to compute dominant eigenvectors?
One-Pass Spectral Sparsification

- **Generalized power iterations with a random vector:**

  \[
  h_0 = \sum_{i=1}^{n} \alpha_i u_i 
  \]

  \[
  \Rightarrow h_t = \left( L_P^+ L_G \right)^t h_0 = \sum_{i=1}^{n} \alpha_i \lambda_i^t u_i 
  \]

- **Embed generalized eigenvalues onto off-tree edges**

  \[
  \Rightarrow h_t^T \delta L_P h_t = \sum_{(p,q) \in G \setminus P} w_{p,q} \sum_{i=1}^{n} \left( \alpha_i \lambda_i \right)^{2t} \left[ u_i^T (e_p - e_q) \right]^2 
  \]

- **Eigenvalue expansion of the Laplacian quadratic form:**

  \[
  \delta L_{P,\text{max}} = L_G - L_P 
  \]

  \[
  \Rightarrow h_t^T \delta L_{P,\text{max}} h_t = \sum_{i=1}^{n} \left( \alpha_i \lambda_i \right)^{2t} (\lambda_i - 1) 
  \]

  Edges with larger Joule heat are more spectrally critical!
Nearly-Linear Time Sparsification Algorithm

1. low-stretch spanning tree extraction
2. t-step generalized power iterations
3. spectral embedding for off-tree edges
4. off-tree edges selection and inclusion

\[ O(m \log n) \text{ time algorithm (prior work)} \]
\[ O(m) \text{ time algorithm (this work)} \]

\( m: \# \text{ of edges} \)
\( n: \# \text{ of nodes} \)
Connection to Effective Resistance Metric

- **Spectrally unique off-tree edge only impact one eigenvalue:**

\[
\Rightarrow e_{p_i} - e_{q_i} = \gamma_i L_p u_i \Rightarrow u_j^T (e_{p_i} - e_{q_i}) = \begin{cases} 
\gamma_i, i = j \\
0, i \neq j
\end{cases}
\]

The effective resistance of edge \((p_i, q_i)\) in sparsifier \(P\):

\[
\Rightarrow R_{p_i,q_i}^\text{eff} = (e_{p_i} - e_{q_i})^T L_p^+ (e_{p_i} - e_{q_i}) = \gamma_i^2
\]

The quadratic form w/ top \(k\) spectrally unique off-tree edges becomes:

\[
h_t^T \delta L_{P,\max} h_t \approx \sum_{i=1}^{k} \alpha_i^2 (\lambda_i)^{2t} w_{p_i,q_i}^T R_{p_i,q_i}^\text{eff} = \sum_{i=1}^{k} \alpha_i^2 (\lambda_i)^{2t} (\lambda_i - 1)
\]

where \(k \ll m\)

**Note:** the sampling probability in [1] is embedded into edge Joule heat

Rank-1 Update of Eigenvalues

- Matrix $A_P = L_G^{1/2} L_P^{1/2} L_G^{1/2}$ has the same eigenvalues as $A = L_P L_G$

- Sherman-Morrison Formula and Matrix Determinant Lemma lead to:

$$A_P^{\text{new}} = L_G^{1/2} \left[ L_P + \mathbf{w}_{p_i, q_i} (e_{p_i} - e_{q_i}) (e_{p_i} - e_{q_i})^T \right]^{+} L_G^{1/2}$$

Eigenvalue reduction $\propto$ the edge sampling probability [1]:

$$\frac{\lambda_i^{\text{old}}}{\lambda_i^{\text{new}}} \approx \mathbf{w}_{p_i, q_i} R_{p_i, q_i}^{\text{eff}}$$

Filtering of Off-Tree Edges

- Quadratic form with worst-case generalized eigenvalues

\[
h_t^T \delta L_{P, \max} h_t \approx \sum_{i=1}^{k} \alpha_i^2 \left( \lambda_i \right)^{2t} \left( \lambda_i - 1 \right) \approx \sum_{i=1}^{k} \alpha_i^2 \left[ \frac{\text{st}_P (G)}{i} \right]^{2t+1}, \text{ with } k \ll m
\]

- Needs \( k \leq O \left( \frac{m \log n \log \log n}{\sigma^2} \right) \) off-tree edges to satisfy \( \lambda_{\text{final}} \leq \sigma^2 \)[1]

- Threshold for filtering off-tree edges:

\[
\theta_k = \frac{\text{Joule}_{\text{Edge}_k}}{\text{Joule}_{\text{Edge}_1}} \propto \left[ \frac{\sigma^2}{\text{st}_P (G)} \right]^{2t+1} \approx \left[ \frac{\sigma^2}{\lambda_{\text{max}}} \right]^{2t+1}
\]

Results of Spectral Edge Embedding

- **Spectral embedding for off-tree edges**
  - Sparse SDD matrices from UFL sparse matrix collection
    - G2\_matrix is for circuit simulation
    - Thermal1 is for unstructured FEM analysis

Not too many large generalized eigenvalues!
Effectiveness of Spectral Graph Sparsification

- 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!

$\sqrt{n} \times \sqrt{n}$ 2D mesh

Initial Joule Heat Map

Final Joule Heat Map

Total Stretch: $O(n\sqrt{n})$
First Few Eigenvectors of Line Graph

"Resonant Frequency"

\[ \tau_1 = 0 \]
\[ \tau_2 > \tau_1 \]
\[ \tau_3 > \tau_2 \]
\[ \tau_4 > \tau_3 \]
Spectral Graph Drawing

- Eigenvectors of unweighted graph Laplacian: $Lv = \tau v$

$$v(p) = \left( \frac{1}{d_p - \tau} \right) \sum_{(p,q) \in E} v(q), \text{ for all } p$$

For small eigenvalues: $v(p)$ is always smooth across neighbors

- Graph drawing using Laplacian eigenvectors
  - Graph embedding using first few nontrivial eigenvectors
    - Example: use $v_2$ and $v_3$ to embed graph into 2D plane
    - Spectral drawing produces optimal layout for grid-like graphs [1]
  - Challenge: too expensive for large graphs when computing eigenvectors
  - Opportunity: drawing sparsified graphs are much easier!
    - 2,047s=>$36s$ (60X faster) for a social network graph (coAuthorsDBLP)

Spectral Graph Drawing

- 2D drawing of Airfoil graph using 2 eigenvectors (v2,v3)
  - 2D coordinates determined by eigenvectors
  - Sparsified graphs always lead to similar drawing results
Spectral Sparsification for Graph Drawing

- 2D drawing of Airfoil graph using 2 eigenvectors (v4, v5)

Original Spectral Drawing

Sparsified Spectral Drawing
Spectral Sparsification for Graph Drawing

- 2D drawing of Airfoil graph using 2 eigenvectors (v6,v7)
Spectral Sparsification for Graph Drawing

- 3D drawing of Airfoil graph using 3 eigenvectors (v2, v3, v4)
Spectral Graph Partitioning and Data Clustering

- Laplacian quadratic forms measure the boundaries of sets:

  \[ x(u) = \begin{cases} 
  1 & \text{if node } u \text{ is in } S \\
  0 & \text{otherwise} 
  \end{cases} \]

  Edges going out \( S \) (boundary):

  \[ x^T L_G x = \text{boundary}(S) \]

- Minimize quadratic form \( \Rightarrow \) minimize boundary (cut) size
  - Basic idea of spectral graph partitioning and data clustering [1]
  - \( k \)-dimensional graph embedding using first \( k \) nontrivial eigenvectors
    - Fiedler vector (the first nontrivial eigenvector) for 1-D embedding
      \[ L_G \nu = \tau \nu \]

Spectral Sparsification for Partitioning & Clustering

- Nearly-linear time graph partitioning/clustering algorithms
  - 1. Extract nearly-linear sized spectral sparsifiers
  - 2. Do partitioning & data clustering on the spectral sparsifiers
Fast Spectral Graph Partitioning Algorithm

- **Spectral graph partitioning using Fiedler vectors (1-D embedding)**
  - The Fiedler vector is computed by a few Inverse Power Iterations (IPI)
  - Each IPI needs to solve a graph Laplacian problem:
    \[ L_G v^k = v^{k-1}, \text{st: } v^k \perp \text{1, for all } k \]
  - Spectral sparsifier allows computing Fiedler vector in almost linear time

**Fiedler vector for a 2D random graph (1000X1000)**

**Solution**

**Error**

Runtime: 12s => 0.5s, 24X speedups!
Big Graph (Data) Compression & Analysis

- **Spectral sparsification for big graph (data) clustering**
  - Over 50X compression (25M=>0.5M) for the RCV1 data set (103 clusters)

![Diagram](image.png)
Sparsification of Nearest Neighbor (NN) Graphs

10-NN graph of USPS data set

Spanning Tree

\[ \lambda_{\text{max}} (L_P^+ L_G) \approx 90,000 \]

1.2-NN Graph

\[ \lambda_{\text{max}} (L_P^+ L_G) \approx 200 \]
Spectral Sparsifiers for Solving Sparse Matrices

- PCG with graph sparsifiers as preconditioners
  - Solve $L_P^+L_Gx = L_P^+b$ instead
  - Smaller relative condition number means faster convergence

Relative Condition Number:

$$\kappa(L_P^+L_G) = \frac{\lambda_{\max}(L_P^+L_G)}{\lambda_{\min}(L_P^+L_G)} = \sigma^2$$

Figure: Simulating Pressure over Airfoil. Source: http://www.nada.kth.se
Results of the PCG Solver (Unstructured Problems)

- **Solving matrices from UFL collection**
  - 5~10% (w.r.t # of nodes) off-tree edges added to the spanning tree
  - Relative residual reduced to $1e-3$; $\lambda_{\text{max}}$ reduced to 100~200
  - Almost linear cost for PCG: less than 1 second & 60 MB per million non-zeros
  - 4-10X speedups & 4-7X memory reduction

<table>
<thead>
<tr>
<th>Test Cases</th>
<th># of Nodes</th>
<th># of Non-zeros</th>
<th>Direct Method (Cholmod)</th>
<th>Iterative Method (PCG)</th>
<th>$\lambda_{\text{max}}$ Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>G3_circuit</td>
<td>1.6E6</td>
<td>7.7E6</td>
<td>45s (2.2G)</td>
<td>8% 5.2s (315M)</td>
<td>45,897X</td>
</tr>
<tr>
<td>thermal2</td>
<td>1.2E6</td>
<td>8.6E6</td>
<td>16.0s (0.9G)</td>
<td>10% 4.4s (235M)</td>
<td>1,582X</td>
</tr>
<tr>
<td>ecology2</td>
<td>1.0E6</td>
<td>5.0E6</td>
<td>12.5s (0.7G)</td>
<td>8% 3.6s (183M)</td>
<td>1,728X</td>
</tr>
<tr>
<td>tmt_sym</td>
<td>0.7E6</td>
<td>5.1E6</td>
<td>11.0s (599M)</td>
<td>10% 2.2s (136M)</td>
<td>796X</td>
</tr>
<tr>
<td>paraboli_fem</td>
<td>0.5E6</td>
<td>3.7E6</td>
<td>6.3s (481M)</td>
<td>8% 1.2s (97M)</td>
<td>120X</td>
</tr>
</tbody>
</table>
Results of the PCG Solver (Resistor Networks)

### Solving power grid analysis matrices
- 2% (w.r.t # of nodes) off-tree edges added to the spanning tree
- Relative residual reduced to $1e-3$; $\lambda_{\text{max}}$ reduced to ~200
- Almost linear cost for PCG: less than 1 second & 60 MB per million non-zeros
- More than 7X speedups and 5X memory reduction

<table>
<thead>
<tr>
<th>CKTs</th>
<th># of Nodes</th>
<th># of Non-zeros</th>
<th>Direct Method (Cholmod)</th>
<th>Iterative Method (PCG)</th>
<th>$\lambda_{\text{max}}$ Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thupg1</td>
<td>5.0E6</td>
<td>2.1E7</td>
<td>75s (4.0G)</td>
<td>27 iter.</td>
<td>10s (0.8G)</td>
</tr>
<tr>
<td>Thupg2</td>
<td>8.9E6</td>
<td>3.9E7</td>
<td>158s (7.6G)</td>
<td>32 iter.</td>
<td>21s (1.5G)</td>
</tr>
<tr>
<td>Thupg3</td>
<td>11.8E6</td>
<td>5.1E7</td>
<td>250s (10.0G)</td>
<td>32 iter.</td>
<td>25s (1.9G)</td>
</tr>
<tr>
<td>Thupg4</td>
<td>15.2E6</td>
<td>6.6E7</td>
<td>N/A</td>
<td>32 iter.</td>
<td>36s (2.5G)</td>
</tr>
<tr>
<td>Thupg5</td>
<td>19.2E6</td>
<td>8.5E7</td>
<td>N/A</td>
<td>33 iter.</td>
<td>47s (3.1G)</td>
</tr>
<tr>
<td>Thupg8</td>
<td>40.0E6</td>
<td>1.8E8</td>
<td>N/A</td>
<td>34 iter.</td>
<td>110s (6.5G)</td>
</tr>
</tbody>
</table>
### Results of the PCG Solver (3D Problems)

- **Solving Laplacians of 3D grids**
  - 5% (w.r.t # of nodes) off-tree edges added to the spanning tree
  - Relative residual reduced to $1\text{e}-3$; $\lambda_{\text{max}}$ reduced to ~500
  - Almost linear cost for PCG: less than 1 second & 60 MB per million non-zeros
  - Up to 162X speedups & 40X memory reduction

<table>
<thead>
<tr>
<th>3D Grid Dim.</th>
<th># of Nodes</th>
<th># of Non-zeros</th>
<th>Direct Method (Cholmod)</th>
<th>Iterative Method (PCG)</th>
<th>$\lambda_{\text{max}}$ Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>50X50X25</td>
<td>6.3E4</td>
<td>4.3E5</td>
<td>4s (240M)</td>
<td>44 iter. 5% 0.5s (16M)</td>
<td>544X</td>
</tr>
<tr>
<td>50X50X50</td>
<td>1.3E5</td>
<td>8.6E5</td>
<td>19s (810M)</td>
<td>46 iter. 5% 1.2s (35M)</td>
<td>443X</td>
</tr>
<tr>
<td>100X50X50</td>
<td>2.5E5</td>
<td>1.7E6</td>
<td>67s (2G)</td>
<td>46 iter. 5% 2.5s (70M)</td>
<td>569X</td>
</tr>
<tr>
<td>100X100X50</td>
<td>0.5E6</td>
<td>3.5E6</td>
<td>301s (5G)</td>
<td>50 iter. 5% 5.8s (160M)</td>
<td>468X</td>
</tr>
<tr>
<td>100X100X100</td>
<td>1.0E6</td>
<td>7.0E6</td>
<td>2,095s (18G)</td>
<td>50 iter. 5% 13s (380M)</td>
<td>515X</td>
</tr>
</tbody>
</table>
Parallel-Friendly PDE Solvers

- **Algebraic multigrid (AMG) solver (e.g. ANSYS Fluent)**
  - Pros: linear-time for solving sparse matrices from elliptic PDEs

- **Setup phase of a recent graph-theoretic AMG method**
  - Lean Algebraic Multigrid (LAMG) (O. Livne et al., SIAM’12)
    - Node elimination phase: eliminate low degree nodes
    - Node aggregation phase: aggregate similar (nearby) nodes
  - Challenge: increasing graph densities at coarse levels during setup
    - High communication cost during parallel solution update

![Node elimination & aggregation](image)
Preservation of Long-Range Effects

- Spectral sparsifiers can well preserve long-range effects
  - More like a “low-pass filter” on the graph
  - Ideal for creating multilevel grid hierarchies

Solution: \( L_G x = b \) (original Laplacian)

Error: \( x - \tilde{x}, \text{where } L_p \tilde{x} = b \) (sparsified Laplacian)
Parallel-Friendly PDE Solvers

- **Sparsified algebraic multigrid (SAMG) solver [1]**
  - SAMG achieves robust convergence and 2X speedups on avg.
  - Dramatically reduced communication cost for parallel processing

### Coarse Level Graph Densities in AMG

**Hierarchy level of 3D_thermal**

<table>
<thead>
<tr>
<th>Level</th>
<th>Graph Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8.9</td>
</tr>
<tr>
<td>2</td>
<td>11.62</td>
</tr>
<tr>
<td>3</td>
<td>13.65</td>
</tr>
<tr>
<td>4</td>
<td>15.13</td>
</tr>
<tr>
<td>5</td>
<td>16.04</td>
</tr>
<tr>
<td>6</td>
<td>16.73</td>
</tr>
<tr>
<td>7</td>
<td>16.18</td>
</tr>
<tr>
<td>8</td>
<td>15.57</td>
</tr>
<tr>
<td>9</td>
<td>16.18</td>
</tr>
<tr>
<td>10</td>
<td>15.75</td>
</tr>
<tr>
<td>11</td>
<td>14.78</td>
</tr>
<tr>
<td>12</td>
<td>13.81</td>
</tr>
<tr>
<td>13</td>
<td>12.74</td>
</tr>
</tbody>
</table>

**Note:** 3D_thermal is mesh grid for 3D thermal analysis

- **SAMG**
- **LAMG**

**Legend:**
- Blue line: w/o spectral sparsification
- Red line: w/ spectral sparsification

**Action:**
- Do sparsification on level 3
Our Ongoing Work

- **Scalable VLSI design & CAD methodologies**
  - Heterogeneous parallel algorithms for EDA
  - Cross-layer, multilevel IC design & verification methods

- **Scalable numerical & graph algorithms**
  - Hardware accelerators for graph-related algorithms
  - Parallel graph-theoretic sparse matrix (PDE) solvers

- **Research goals**
  - Big chip design automation on a personal computer
  - Big data (graph) processing on a hand-held device
Questions?

Hawaiian Lava Flow Out of Michigan Arctic Snow

Figure Source: MTU Winter Carnival 2015