Specialized System Solvers for very large Systems; Theory and Practice

Gary Miller

Carnegie Mellon University
join work with Yiannis Koutis, Richard Peng, Ali Sinop, and David Tolliver

Workshops on Algorithms for Modern Massive Data Sets
June 18, 2010

Outline

1. Introduction
2. Symmetric Diagonally Dominate Systems
   - Graph Laplacians
3. Applications of SDD/Laplacians
4. Iterative Methods for Graph Laplacians
   - Combinatorial Preconditioners
5. Low Stretch Spanning Trees
6. Solver Code
7. Open Questions

Solving Linear Systems, a fundamental Problem

$$\begin{pmatrix} 3 & 2 & -1 \\ 2 & -5 & 4 \\ -1 & 1/2 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 3 \\ 7 \\ 2 \end{pmatrix}$$

An Easy Case

1. Upper and Lower Triangular Systems

An Easy Case

1. Upper and Lower Triangular Systems
2. $O(m)$ time where $m =$ number of nonzeros entries.

An Easy Case

1. Upper and Lower Triangular Systems
2. $O(m)$ time where $m =$ number of nonzeros entries.
3. Goal: Find more easy cases that have applications.

Symmetric Matrices and Positive Definite

- Assume A is symmetric, $A = A^T$.

Symmetric Matrices and Positive Definite

- Assume A is symmetric, $A = A^T$.
- Assume A is positive definite, $x^T A x > 0$ for $x \neq 0$.

Symmetric Matrices and Positive Definite

- Assume A is symmetric, $A = A^T$.
- Assume A is positive definite, $x^T Ax > 0$ for $x \neq 0$.
- Open: **Can we solve spd systems in near linear time?**

Symmetric Matrices and Positive Definite

- Assume A is symmetric, $A = A^T$.
- Assume A is positive definite, $x^T A x > 0$ for $x \neq 0$.
- Open: **Can we solve spd systems in near linear time?**
- Two approaches to solving: direct and iterative methods.

Direct Methods
Gaussian Elimination Matrices

- Goal: algorithms that minimize work and space.

Direct Methods
Gaussian Elimination Matrices

- Goal: algorithms that minimize work and space.
- Trick: View nonzero entries as an undirected graph and view pivoting as a graph operation.

Good Pivot Strategies

1970s and 1980s

- Planar systems: $O(n^{3/2})$ work and $O(n \log n)$ fill/space, [Lipton, Rose, Tarjan].

Good Pivot Strategies

1970s and 1980s

- Planar systems: $O(n^{3/2})$ work and $O(n \log n)$ fill/space, [Lipton, Rose, Tarjan].
- 3D Systems: $O(n^2)$ work and $O(n^{3/2})$ fill/space, [M, Teng, Thurston, Vavasis] (EG: 3D images and 3D finite element).

Good Pivot Strategies

1970s and 1980s

- Planar systems: $O(n^{3/2})$ work and $O(n \log n)$ fill/space, [Lipton, Rose, Tarjan].
- 3D Systems: $O(n^2)$ work and $O(n^{3/2})$ fill/space, [M, Teng, Thurston, Vavasis] (EG: 3D images and 3D finite element).
- $O(n^{3/2})$ space is too big for 3D Image problems.

Pure Iterative Methods

Solving $Ax = b$.

- Basic method: $x^{(i+1)} = (I - A)x^{(i)} + b$

Pure Iterative Methods

Solving $Ax = b$.

- Basic method: $x^{(i+1)} = (I - A)x^{(i)} + b$
- Convergence/Rate is determined by $||I - A||$.

Pure Iterative Methods

Solving $Ax = b$.

- Basic method: $x^{(i+1)} = (I - A)x^{(i)} + b$
- Convergence/Rate is determined by $||I - A||$.
- Accelerated Methods: Chebyshev Iteration, Conjugate Gradient.

Pure Iterative Methods

Solving $Ax = b$.

- Basic method: $x^{(i+1)} = (I - A)x^{(i)} + b$
- Convergence/Rate is determined by $||I - A||$.
- Accelerated Methods: Chebyshev Iteration, Conjugate Gradient.
- CG: $O(nm)$, [Magnus, Eduard 52].

Preconditioned Iterative Methods

Solving $B^{-1}Ax = B^{-1}b$.

- Basic method: $x^{(i+1)} = x^{(i)} + B^{-1}(b - Ax^{(i)})$

Preconditioned Iterative Methods

Solving $B^{-1}Ax = B^{-1}b$.

- Basic method: $x^{(i+1)} = x^{(i)} + B^{-1}(b - Ax^{(i)})$
- Computing $x^{(i+1)}$
  - $r = b - Ax^{(i)}$ Forward Multiply and addition.
  - $Bz = r$ Solve the preconditioner system
  - return $x^{(i+1)} = x^{(i)} + z$

Preconditioned Iterative Methods

Solving $B^{-1}Ax = B^{-1}b$.

- Basic method: $x^{(i+1)} = x^{(i)} + B^{-1}(b - Ax^{(i)})$
- Computing $x^{(i+1)}$
  - $r = b - Ax^{(i)}$ Forward Multiply and addition.
  - $Bz = r$ Solve the preconditioner system
  - return $x^{(i+1)} = x^{(i)} + z$
- Goal: Minimize the number of iteration while minimizing the cost of the solve.

Classic Preconditioners

- Jacobi: $B = Diagonal(A)$.

Classic Preconditioners

- Jacobi: $B = Diagonal(A)$.
- Gauss-Seidel: $B = UpperTriangular(A)$.

Classic Preconditioners

- Jacobi: $B = Diagonal(A)$.
- Gauss-Seidel: $B = UpperTriangular(A)$.
- SSOR: $B = (L + \frac{1}{\omega}D)\frac{1}{\omega}D(L + \frac{1}{\omega}D)$

Classic Preconditioners

- Jacobi: $B = Diagonal(A)$.
- Gauss-Seidel: $B = UpperTriangular(A)$.
- SSOR: $B = (L + \frac{1}{\omega}D)\frac{1}{\omega}D(L + \frac{1}{\omega}D)$
- Still too slow and unreliable.

Symmetric Diagonally Dominate Matrices

- Def: A is SDD if:

$$\forall i \quad A_{ii} \geq \sum_{j \neq i} |A_{ij}|$$

Symmetric Diagonally Dominate Matrices

- Def: A is SDD if:

$$\forall i \quad A_{ii} \geq \sum_{j \neq i} |A_{ij}|$$

- Note: A is positive semi-definite.

Symmetric Diagonally Dominate Matrices

- Def: A is SDD if:

$$\forall i \quad A_{ii} \geq \sum_{j \neq i} |A_{ij}|$$

- Note: A is positive semi-definite.
- Subcase: SDD with nonpositive off diagonal
  Graph Laplacians

Symmetric Diagonally Dominate Matrices

- Def: A is SDD if:

$$\forall i \quad A_{ii} \geq \sum_{j \neq i} |A_{ij}|$$

- Note: A is positive semi-definite.
- Subcase: SDD with nonpositive off diagonal
  Graph Laplacians
- SDD can be reduce to Graph Laplacians, [Gremban M 96]

Graph Laplacian

- $G = (V, E, w)$ weighted undirected graph, $w_{ij} > 0$.

Graph Laplacian

- $G = (V, E, w)$ weighted undirected graph, $w_{ij} > 0$.
- Weighted incidence matrix:

$$A_{ij} = \begin{cases} w_{ij} & \text{if } e_{ij} \in E \\ 0 & \text{otherwise} \end{cases}$$

Graph Laplacian

- $G = (V, E, w)$ weighted undirected graph, $w_{ij} > 0$.
- Weighted incidence matrix:

$$A_{ij} = \begin{cases} w_{ij} & \text{if } e_{ij} \in E \\ 0 & \text{otherwise} \end{cases}$$

- Degree of v_i: $d_i = \sum_j w_{ij}$

Graph Laplacian

- $G = (V, E, w)$ weighted undirected graph, $w_{ij} > 0$.
- Weighted incidence matrix:

$$A_{ij} = \begin{cases} w_{ij} & \text{if } e_{ij} \in E \\ 0 & \text{otherwise} \end{cases}$$

- Degree of v_i: $d_i = \sum_j w_{ij}$
-

$$D = \begin{pmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_n \end{pmatrix}$$

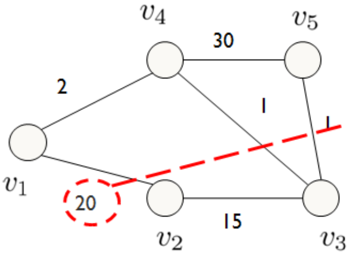
Graph Laplacian

- $G = (V, E, w)$ weighted undirected graph, $w_{ij} > 0$.
- Weighted incidence matrix:

$$A_{ij} = \begin{cases} w_{ij} & \text{if } e_{ij} \in E \\ 0 & \text{otherwise} \end{cases}$$

- Degree of v_i: $d_i = \sum_j w_{ij}$
- $$D = \begin{pmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_n \end{pmatrix}$$
- Laplacian: $L = D - A$

Example of Laplacian



$$L = \begin{pmatrix} 22 & -20 & 0 & -2 & 0 \\ -20 & 35 & -15 & 0 & 0 \\ 0 & -15 & 17 & -1 & -1 \\ -2 & 0 & -1 & 33 & -30 \\ 0 & 0 & -1 & -30 & 31 \end{pmatrix}$$

History for solving Laplacian

Recurrsive preconditioned iterative methods.

- $O(n^{1.2})$ for planar Laplacians, [Vaidya 91]
- $\tilde{O}(m^{1.5})$ for natural 3D graphs [Gremban, M 96].
- First near-linear time algorithm, $O(m \log^{15} n)$, [Spielman, Teng 04].
- $O(n)$ for planar Laplacians, [Koutis, M 07]
- $O(m \log^2 n)$ (ignoring $\log\log$ and lower terms), [Koutis, M, Peng 10].

Main Theorem

Theorem

Input:	*SDD system* $Ax = b$.
Output:	$\bar{x}$ *satisfying* $\|\|\bar{x} - A^{+}b\|\|_A < \epsilon \|\|A^{+}b\|\|_A$.
Expected Time:	$\tilde{O}(m \log^2 n \log(1/\epsilon))$.

[Koutis, M, Peng 10]

Classic Applications of the Laplacian

- View each edge a conductor with conductance w_{ij}.

Classic Applications of the Laplacian

- View each edge a conductor with conductance w_{ij}.
- Let V be a column vector of voltages

Classic Applications of the Laplacian

- View each edge a conductor with conductance w_{ij}.
- Let V be a column vector of voltages
- If $LV = c$ then c is the residual current needed to maintain the given voltages.

Graph Laplacian's and the Heat Equations

- View each edge as a conductor with conductance w_{ij}.

Graph Laplacian's and the Heat Equations

- View each edge as a conductor with conductance w_{ij}.
- Let V be a column vector of temperatures.

Graph Laplacian's and the Heat Equations

- View each edge as a conductor with conductance w_{ij}.
- Let V be a column vector of temperatures.
- If $c = LV$ then c is the residual heat needed to maintain the given temperatures.

Graph Laplacian's and the Heat Equations

- View each edge as a conductor with conductance w_{ij}.
- Let V be a column vector of temperatures.
- If $c = LV$ then c is the residual heat needed to maintain the given temperatures.
- The finite element heat equations can be preconditioned with a graph Laplacian and thus solved in $\tilde{O}(n + m)$ time. [Boman, Hendrickson, and Vavasis 06]

Graph Laplacian's and Random Walks

Transition Matrix: $A_G D^{-1}$, symmetric A. Mixing Rate-Fundamental Eigenvector:
$\tilde{O}(n+m)$ [Spielman Teng 04]
Trick: Inverse Powering only requires $O(\log n)$ iterations.

Laplacian's and Spring Mass Systems

- $G = (V, E, w)$ weighted graph and w_{ij} is viewed a spring constant.

Laplacian's and Spring Mass Systems

- $G = (V, E, w)$ weighted graph and w_{ij} is viewed a spring constant.
- M is a diagonal matrix of mass constants

Laplacian's and Spring Mass Systems

- $G = (V, E, w)$ weighted graph and w_{ij} is viewed a spring constant.
- M is a diagonal matrix of mass constants
- Fact: Modes of vibration of Spring-Mass system G, M are:
  Eigen-pairs of $L_G x = \lambda M x$.

Laplacian's and Spring Mass Systems

- $G = (V, E, w)$ weighted graph and w_{ij} is viewed a spring constant.
- M is a diagonal matrix of mass constants
- Fact: Modes of vibration of Spring-Mass system G, M are:
  Eigen-pairs of $L_G x = \lambda M x$.
- Thus the fundamental mode can be found in $\tilde{O}(n + m)$ time.

Spring Mass System

Movie of a Simple Image

Graph Laplacian's and Linear Programming

- Graph Maximum Flow Prob: Find a maximum flow from s to t.

Graph Laplacian's and Linear Programming

- Graph Maximum Flow Prob: Find a maximum flow from s to t.
- Algorithm: Max-Flow is a LP problem so use log barrier interior point method.

Graph Laplacian's and Linear Programming

- Graph Maximum Flow Prob: Find a maximum flow from s to t.
- Algorithm: Max-Flow is a LP problem so use log barrier interior point method.
- Fact: Each of $O(\sqrt{m})$ pivots requires the solution the graph Laplacian.

Graph Laplacian's and Linear Programming

- Graph Maximum Flow Prob: Find a maximum flow from s to t.
- Algorithm: Max-Flow is a LP problem so use log barrier interior point method.
- Fact: Each of $O(\sqrt{m})$ pivots requires the solution the graph Laplacian.
- Thus: Approximate Max-Flow is $\tilde{O}((m+n)^{3/2})$ [Daitch, Spielman 08]

Graph Laplacian's and Convex Programming

- Uniform TV Denoising:
  Input: image s
  Output: image $\arg\min ||x - s||_2^2 + \lambda|| \bigtriangledown x||_1$

Graph Laplacian's and Convex Programming

- Uniform TV Denoising:
  Input: image s
  Output: image $\arg\min ||x - s||_2^2 + \lambda|| \bigtriangledown x||_1$
- Nonuniform TV Denoising:
  Input: pixel image s
  Output: $\arg\min (x - s)^T (x - s) + Sum_{(i,j)\in G} |w_{ij}(x_i - x_j)|$

Graph Laplacian's and Convex Programming

- Uniform TV Denoising:
  Input: image s
  Output: image $\arg\min ||x - s||_2^2 + \lambda || \bigtriangledown x||_1$
- Nonuniform TV Denoising:
  Input: pixel image s
  Output: $\arg\min (x - s)^T (x - s) + Sum_{(i,j)\in G} |w_{ij}(x_i - x_j)|$
- Use log-barrier interior point: pivots are low rank perturbation of Laplacian,
  Thus: $\tilde{O}((m + n)^{3/2})$ time. [Koutis M Peng Sinop Tolliver 09]

Condition Number

- **Def:** Condition number of A and B, $\kappa(B^{-1}A) = \lambda_{max}(B^{-1}A)/\lambda_{min}(B^{-1}A)$.
- OR: If $x^TAx \leq x^TBx \leq kx^TAx$ for all $x \in \mathbb{R}^n$, then $\kappa(B^{-1}A) \leq k$

Rate of Convergence

- Classical results, measured in number of iterations per bit of precision.
- Richardson iteration: $O(\kappa(B^{-1}A))$, too slow.
- Conjugate gradient: $O(\sqrt{\kappa(B^{-1}A)})$ or better, hard to analyze when B is called recursively and solved inexactly.
- Chebyshev iteration: $O(\sqrt{\kappa(B^{-1}A)})$, will use.

When is B a Good Recursive Preconditioner?

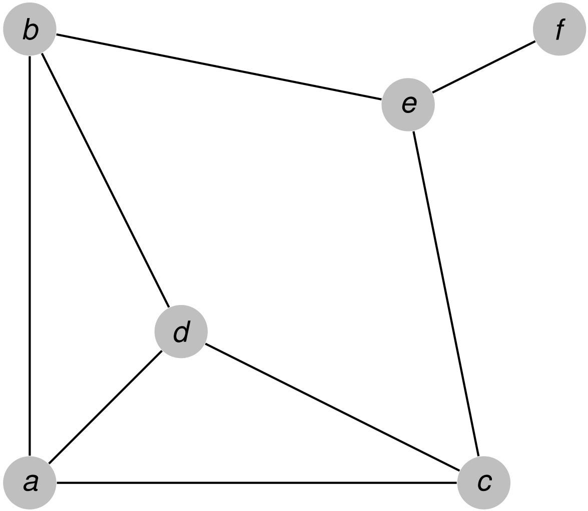
- Properties that Laplacian B should have:
  1. $B^{-1}A$ has low condition number.
  2. Quickly reduces to something that can be solved faster (smaller size).
- Examples:
  - [Vaidya 91] Spanning tree + a few edges.
  - [Gremban, M 96] Steiner tree.
  - [Boman, Hendrickson 03; Spielman, Teng 04] Low stretch spanning tree + a few edges.
  - [Koutis, M 07] Partition planar graphs into pieces of size k with $\sqrt{k}$ boundary, optimally precondition each piece.

Getting a Good Preconditioner.

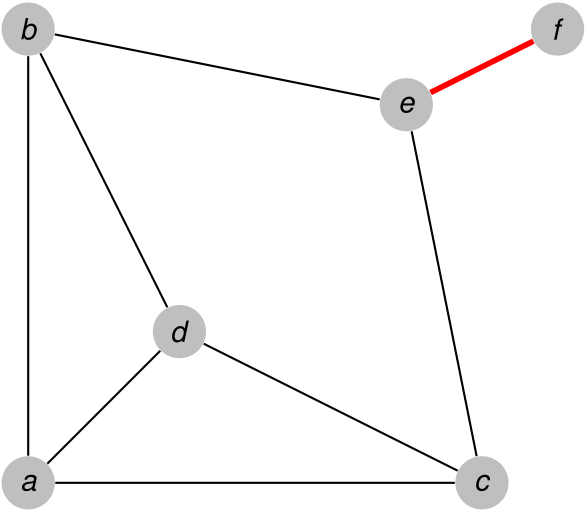
Main steps:

- Find a sparse subgraph by random sampling.
- Use Gaussian elimination to remove degree 1 and 2 vertices.
- We need sampling to be fast and give good condition numbers.

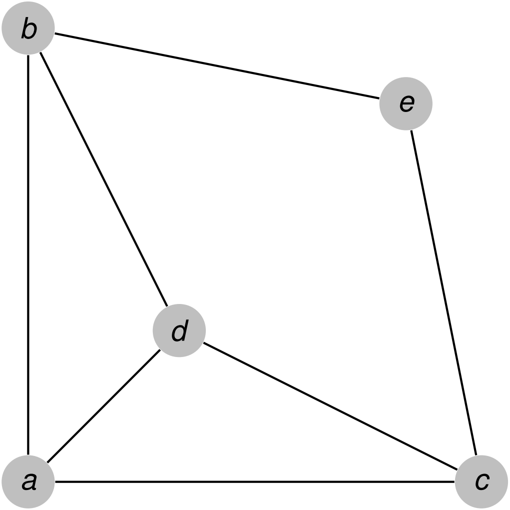
Example: Pivoting out degree 1 and 2.



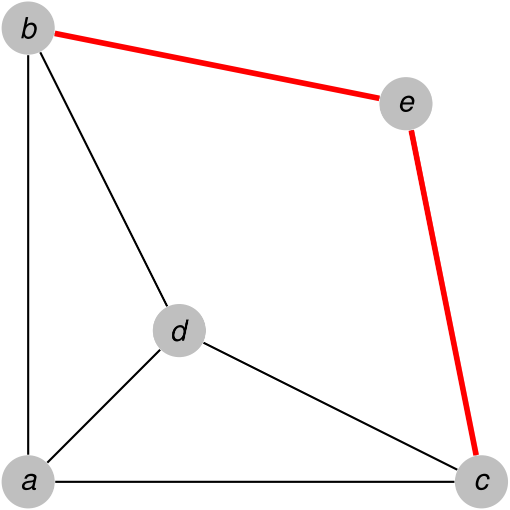
Example: Pivoting out degree 1 and 2.



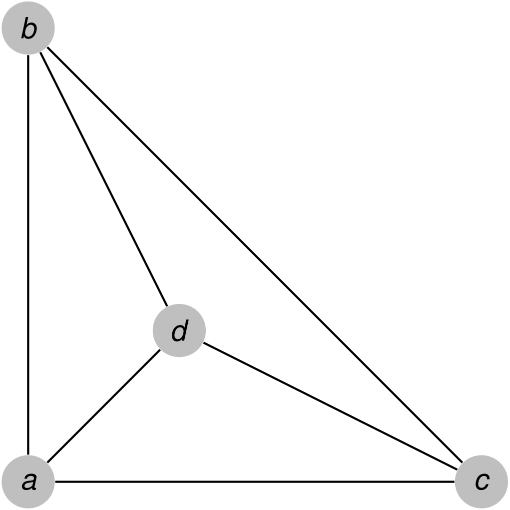
Pivot(f)



Pivot(f)



Pivot(e)



Ran out of degree 1 or 2 nodes, quit.

Graph Sparsifier

Given a graph G find H by: Remove most edges, increase weight of remaining edges.
Possible properties to be preserved:

- Spanners: distance, diameter
- Cut sparsifier: weight of cut for all $2^{|V|}$ subset of vertices
- Triangle sparsifiers: number of triangles in a subgraph
- We want spectral sparsifiers
- Also want H to be ultra-sparse for Gaussian elimination to make progress.
- Need to reduce to $n - 1 + m/c$ edges for in order to decrease edge count by factor of $c/3$.

Previous Work on Sparsification

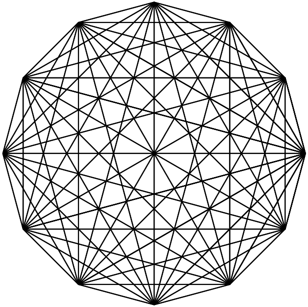
- Expanders: sparsifier for complete graph.
- Ramanujan graphs: optimal spectral sparsifiers for the complete graph.
- [Benczur, Karger 96] Cut sparsifiers, $O(n \log n)$ edges.
- [Kolountzakis, M, Tsourakakis 10] Edge sampling can give good triangle sparsifiers.

Previous Work on Spectral Sparsification

- [Spielman, Teng 04] H has $\tilde{O}(n)$ edges, constant condition number.
- [Spielman, Teng 04] Ultrasparsifier, H has $n - 1 + n/c$ edges, condition number $\tilde{O}(c)$.
- **[Spielman, Srivastava 08] Conceptually simple sampling algorithm for spectral sparsification.**
- [Batson, Spielman, Srivastava 09] and [Kolla, Makarychev, Saberi, Teng 10] gave better bounds, but their algorithms do not run in near-linear time.

Sparsifier for the Complete Graph

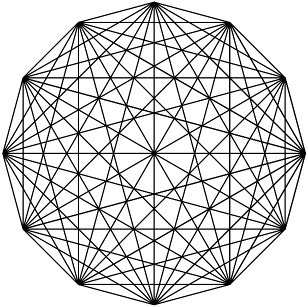
- For K_n, $\frac{1}{p}G_{(n,p)}$ is a good sparsifier when $p \geq \log n/n$.

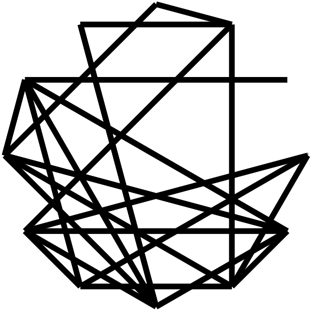


- Sidenote: examples generated by code in the TeX file, different sparsifier every time slides are generated.

Sparsifier for the Complete Graph

- For K_n, $\frac{1}{p}G_{(n,p)}$ is a good sparsifier when $p \geq \log n/n$.





- Sidenote: examples generated by code in the TeX file, different sparsifier every time slides are generated.

Generalized Graph Sampling

- [Benczur, Karger 96] used this method for cut sparsifiers
- Near-linear time spectral sparsifiers use the same framework.
- Compute a probability p_e for each edge.
- For each edge e keep with probability p_e.
  If kept multiply weight by $1/p_e$.

What Sampling Gives

- Expected value: original graph
- Expected number of edges: $(\sum_e p_e) \log n$.
- Concentration?

Effective Resistance

- Consider each edge as a resistor with conductance w_e
- For edge $e = (u, v)$ let R_e be the effective resistance from u to v in G.

Sparsification by Effective Resistance

Theorem (Spielman, Srivastava 08)

Sampling a weighted graph G using edge probablities $p_e = w_e R_e$ to generate H with $O(n \log n)$ expected edges then $\kappa(G, H)$ is a constant with high probablity.

Calculating effective resistance efficiently?

Low Stretch Spanning Trees

We use low stretch spanning trees to approximate effective resistance.
Let T be a tree of G

Definition

Stretch(e) = $w_e \cdot ER_e^T$, the effective resistance in T.

$$stretch(T) = \sum_{e \in G} stretch(e)$$

Low Stretch Spanning Trees

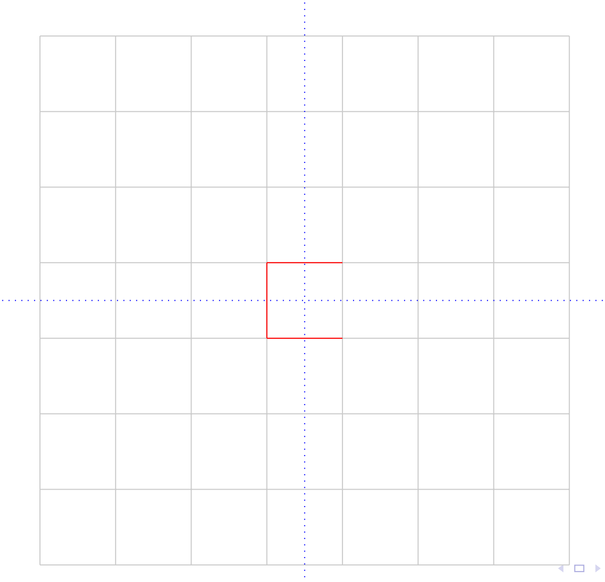
- [Spielman, Teng 04] Fundamental for their solver and ultrasparsifier.
- [Kolla, Makarychev, Saberi, Teng 10] Fundamental their near-optimal sparsifier.

Known Results about Low Stretch Spanning Trees

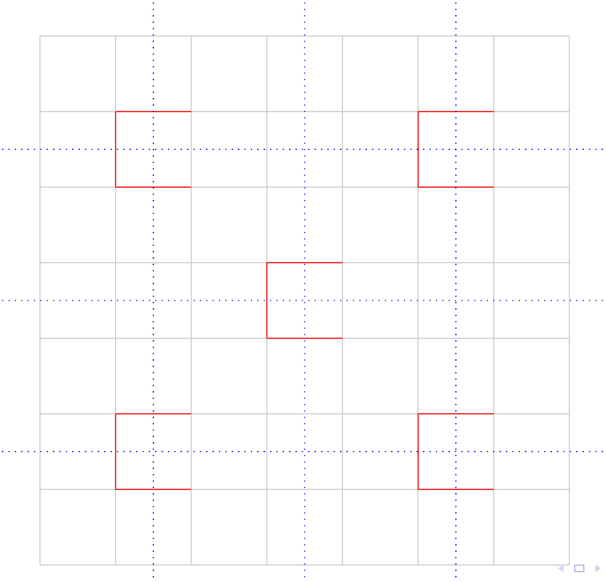
- First studied in [Alon, Karp, Peleg, West 95] in the context of k server problem.
- [Elkin, Emek, Spielman, Teng 05] $O(m \log^2 n)$ stretch.
- [Abraham, Bartal & Neiman 08] roughly $O(m \log n)$ stretch.

Example of Low Stretch Spanning Tree on a Unit Weight Mesh

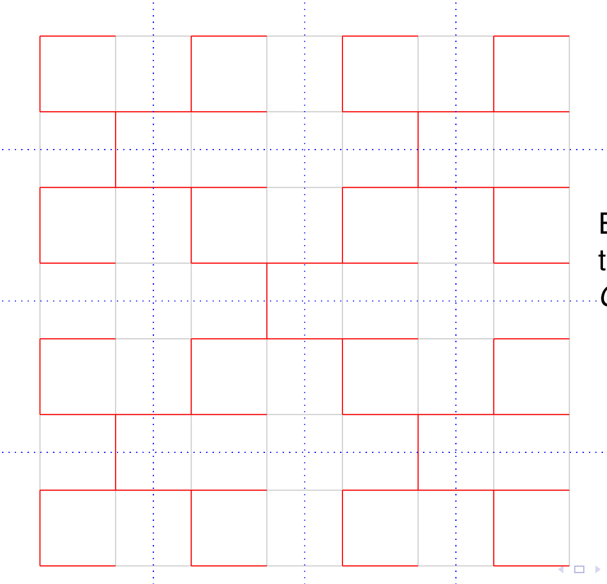
Example of Low Stretch Spanning Tree on a Unit Weight Mesh



Example of Low Stretch Spanning Tree on a Unit Weight Mesh



Example of Low Stretch Spanning Tree on a Unit Weight Mesh



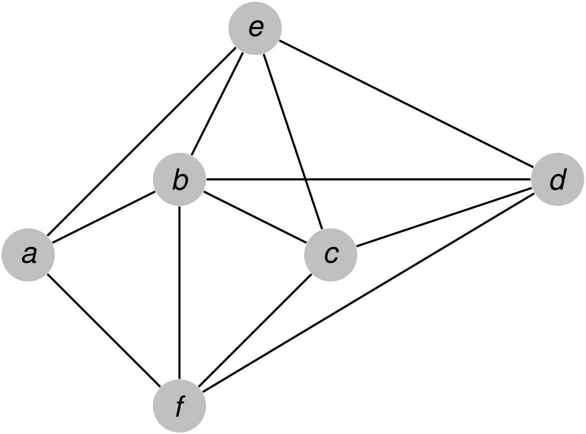
Exercise: show that stretch is $O(n \log n)$.

Pseudocode

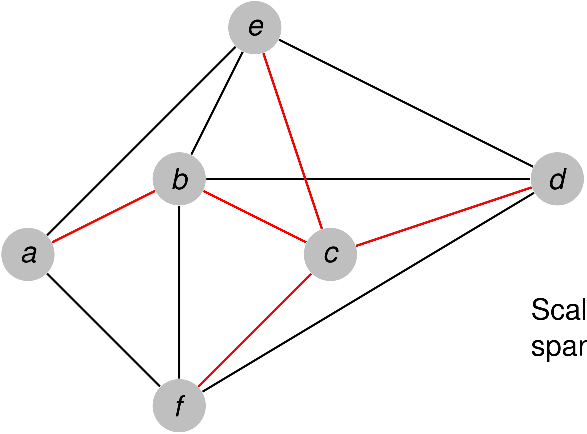
INCREMENTALSPARSIFY
Input: Graph G, real value $c = O(log^4 n)$.
Output: Graph H that's a sparsifier for G

1. $T \leftarrow$ LOWSTRETCHTREE(G)
2. Let T' be T scaled up by factor of c
3. Let G' be the graph obtained from G by replacing T with T'
4. FOR $e \in E$
5. Calculate $EffectiveResistance_{T'}(e)$
6. ENDFOR
7. $H \leftarrow$ SAMPLE(G', $EffectiveResistance_{T'}$)
8. RETURN H

Example: Original Graph

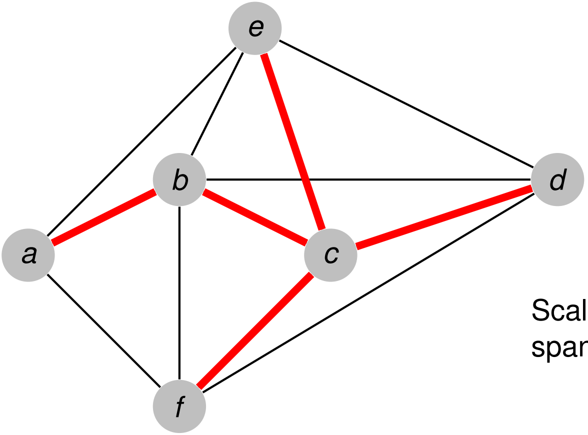


Example: Scale up a Good Spanning Tree



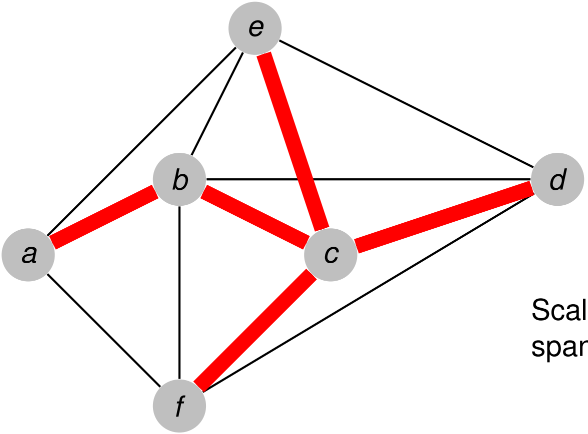
Scale up the spanning tree.

Example: Scale up a Good Spanning Tree



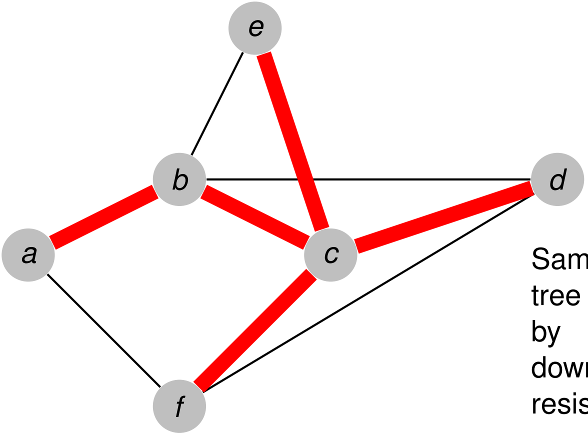
Scale up the spanning tree.

Example: Scale up a Good Spanning Tree



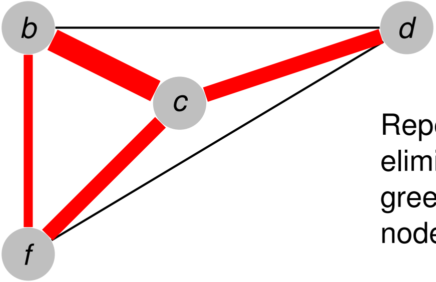
Scale up the spanning tree.

Example: Sample



Sample non-tree edges by (scaled down) effective resistance.

Example: Gaussian Elimination



Repeatedly eliminate degree 1 and 2 nodes.

History of Planar Solvers

- 1950's $O(n^2)$ (Conjugate Gradient)

History of Planar Solvers

- 1950's $O(n^2)$ (Conjugate Gradient)
- 1970's $O(n^{1.5})$ (Nested Dissection) (LRT)

History of Planar Solvers

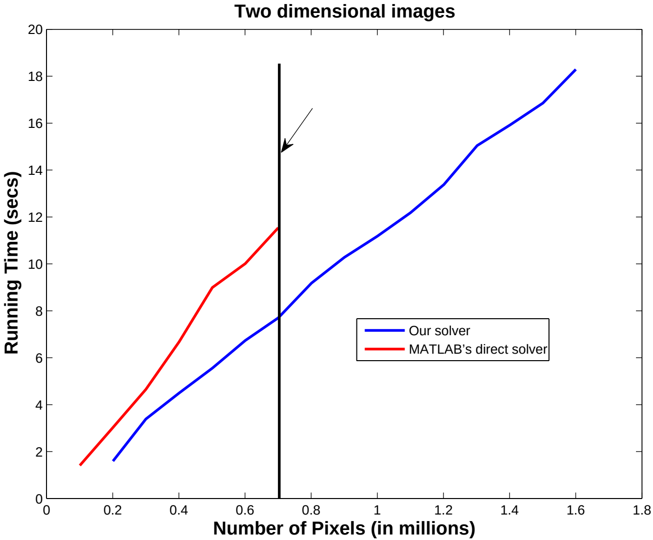
- 1950's $O(n^2)$ (Conjugate Gradient)
- 1970's $O(n^{1.5})$ (Nested Dissection) (LRT)
- 1990's $O(n^{1.2})$ (Combinatorial Preconditioners) (Vaidya)

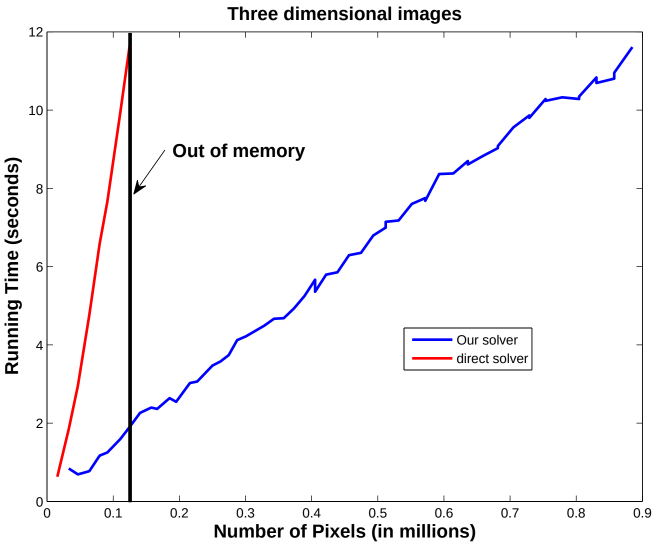
History of Planar Solvers

- 1950's $O(n^2)$ (Conjugate Gradient)
- 1970's $O(n^{1.5})$ (Nested Dissection) (LRT)
- 1990's $O(n^{1.2})$ (Combinatorial Preconditioners) (Vaidya)
- 2000's $O(n \log^2 n)$ (Low stretch spanning trees) (ST)

History of Planar Solvers

- 1950's $O(n^2)$ (Conjugate Gradient)
- 1970's $O(n^{1.5})$ (Nested Dissection) (LRT)
- 1990's $O(n^{1.2})$ (Combinatorial Preconditioners) (Vaidya)
- 2000's $O(n \log^2 n)$ (Low stretch spanning trees) (ST)
- 2006's $O(n)$ (separator based preconditioners) (KM)


Two dimensional images
Running Time (secs)
Number of Pixels (in millions)
Our solver
MATLAB's direct solver
0
2
4
6
8
10
12
14
16
18
20
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8


Three dimensional images
Running Time (seconds)
Number of Pixels (in millions)
0
2
4
6
8
10
12
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Out of memory
Our solver
direct solver

Open Questions

- Find fast methods for any SPD system.
- Find spectral methods that find better cuts by using more than one eigenvector.
- Find solvers that work in the L_2 norm.
- A implementable solver with near linear time guarantees. The low stretch spanning tree is the bottleneck!

Thank You