Yiannis Koutis
University of Puerto Rico, Rio Piedras

joint with **Gary Miller, Richard Peng**
Carnegie Mellon University

SDD Solvers: Bridging theory and practice

National Science Foundation
WHERE DISCOVERIES BEGIN

The problem:
Solving very large **Laplacian** systems

$$A*x = b$$

dimension n, m non-zero entries

- **Laplacian:**
- symmetric, negative off-diagonal elements, zero row-sums

The problem:
Solving very large **Laplacian** systems

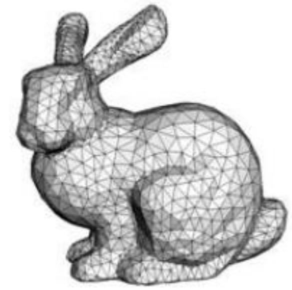
$$* \quad x = b$$

- Will keep algebra to a minimum

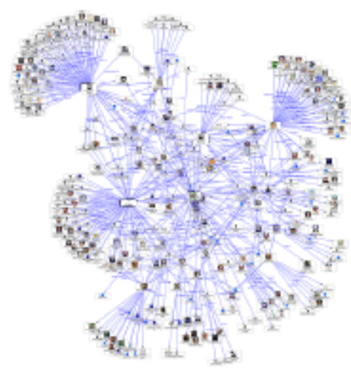
Why solve Laplacians?

It would take a whole book to cover their applications

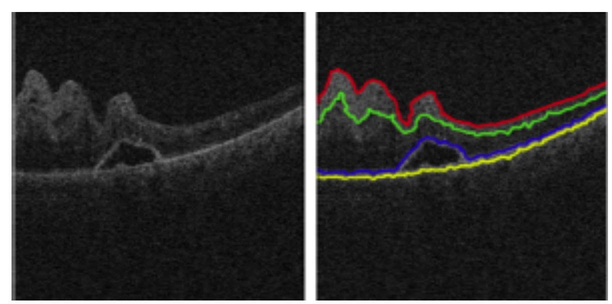
- Traditionally useful in scientific computing
  - Solving problems on nice meshes



- Numerous novel applications in **networks**
  - Link prediction
  - Recommendation systems
  - Protein interaction networks

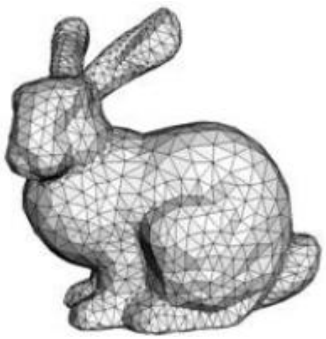


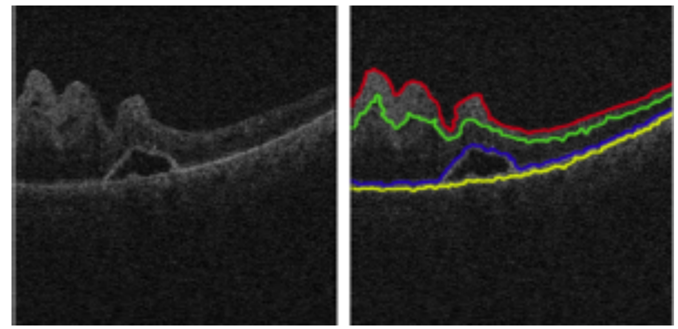
- Several applications in **computer vision**
  - Image denoising
  - Image impainting
  - Image segmentation [KMT09]



What is the reason … ?

Laplacians are so common in applications



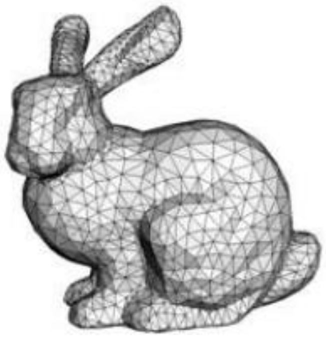


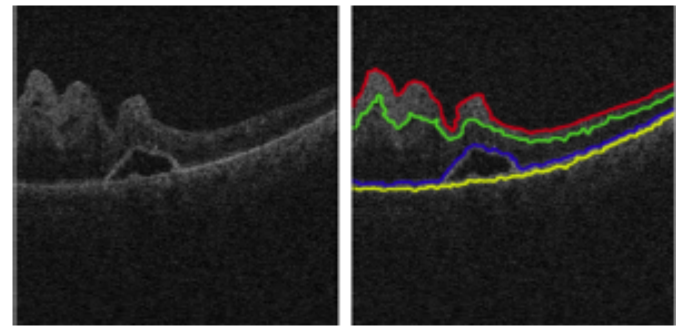
- It's the algebra behind Ohm's and Kirchoff's laws that govern **electrical flows** in resistive networks
- They also describe **random walks** in graphs
- Their eigenvectors capture information about **graph cuts**

- **… so they appear spontaneously**

What is the reason … ?

Laplacians are so common in applications





- **… but also because we've become suspicious about their power**
- In 2004, Spielman and Teng showed that Laplacians can be solved in nearly-linear time $O(m\log^{50} n)$
- Laplacian-based solutions have accelerated since then

The Laplacian paradigm

- With recent improvements Laplacian solvers are increasingly viewed as a powerful **algorithmic primitive**

- **A great example:** The new fastest known algorithm for the long-studied max-flow/min-cut problem is solver-based

"Spielman and Teng have ignited what appears to be an incipient revolution in the theory of graph algorithms"

--Goemans, Kelner

- Shang-Hua Teng: *"The Laplacian paradigm"*
- Erica Klarreich: *"Network Solutions"* (simonsfoundation.org)

The goal of the talk:

- State-of-the-art in **theory :**
- A provably **O(mlog n)** algorithm [Koutis, Miller, Peng 10-11]

- State-of-the-art in **practice :**
- An empirically **linear time** implementation [Koutis, Miller 08]
- ….. with very rare exceptions

- Bridging the gap
- Ideas for **"sketching"** large graphs

Instance sparsification

a basic idea in algorithm design

- Solve a **sparser but similar instance** of the problem
- **Similar** here depends on the application

- Some examples (actually related to our problem too)

Instance sparsification

a basic idea in algorithm design

- **Distance-based** sparsification
- Each graph A contains a **sparse** subgraph B (spanner) so that for every pair of vertices (u,v):

$$d_A(u,v) \leq d_B(u,v) \leq \log n * d_A(u,v)$$

- **Cut-based** sparsification
- For each graph A there is a **sparse** graph B
  where all bipartitions are nearly the same as in A
- Community detection lumps up parts of the graph

Spectral sparsification

sparsification in the context of linear systems

- It is called **spectral** because graphs A and B have about the same eigenvalues and eigenspaces

- Spectral sparsification in some subsumes distance-based **and** cut-based sparsification

- In numerical analysis B is known as the **preconditioner**
- Preconditioning is a very well studied topic mostly as an **algebraic** problem

- Pravin Vaidya saw preconditioning as a **graph** problem

Spectral sparsification

sparsification in the context of linear systems

- Algebra guides us to:
  - Find the appropriate measure of similarity
  - Distill sufficient conditions that lead to a fast solver

- Graph B is an **incremental sparsifier** with two properties

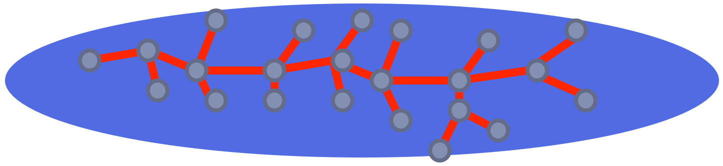
1. $$m_B \leq m_A / \sqrt{\kappa}$$

2. $$x^T A x \leq x^T B x \leq \kappa * x^T A x$$

Quality of approximation

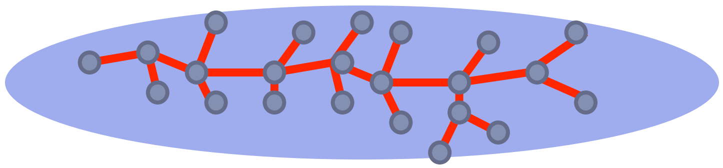
Recursive sparsification:

a construction of **a chain of preconditioners**



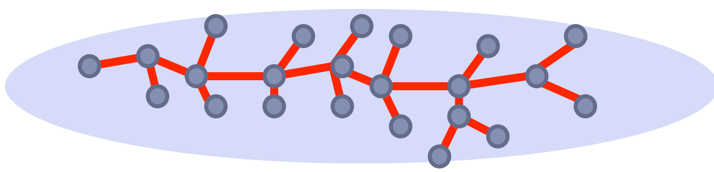
Find a tree:
Multiply weights on tree by $\log^2 n$

Which tree?



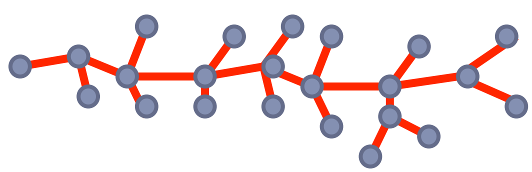
Incremental sparsifier via sampling
Make tree a little heavier

What probabilities?



Recursion

So what?
System is still big



…until left with a heavier tree

The **theory**:

a construction of **a chain of preconditioners**

Which tree ?

- There is a **low-stretch spanning tree** that preserves edge weights within an O(log n) factor **on average**

What probabilities ?

- The probability that an edge is sampled is proportional to its **stretch** over the low-stretch tree. The higher the stretch the more probable is that the edge will be kept in the sparsifier.

- Somehow, preserving distances + randomization amounts to spectral sparsification!

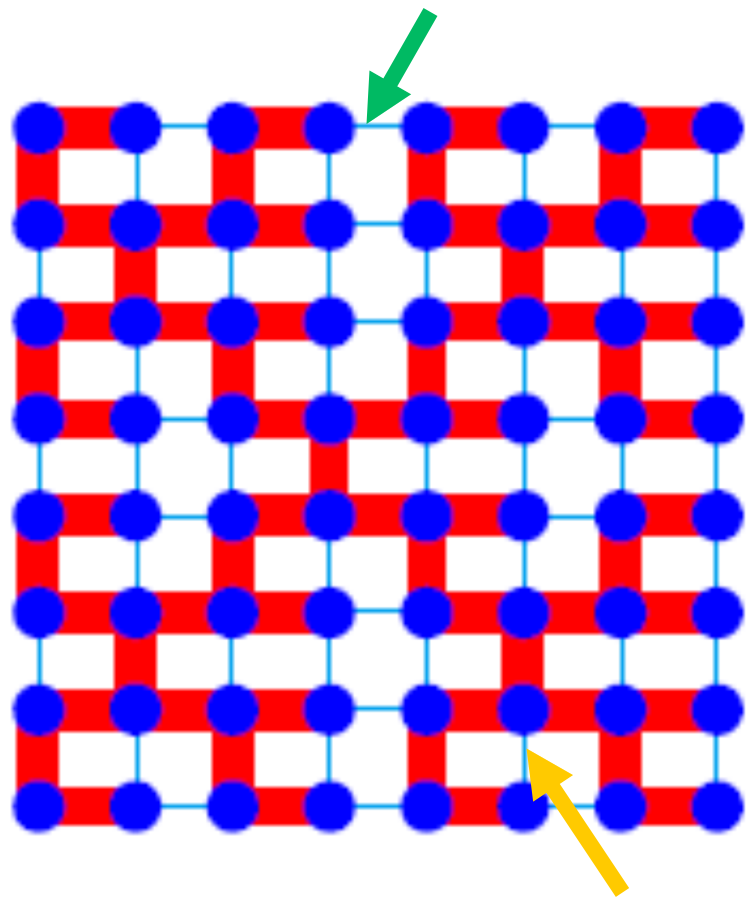
Low-stretch tree

an illustration on the grid

- Most edges have low stretch

- There are edges with high stretch
- Their number is small
- So they don't affect average

- These edges tend to **stay** in the chain until the near-end when tree has become heavy enough to absorb them



The **publicity** ☺

- Including Slashdot, CACM, Techreview (MIT)….

ACM.ORG JOIN ACM

COMMUNICATIONS OF THE ACM

TRUSTED INSIGHTS FOR COMPUTING'S LEADING PROFESSIONALS

A Breakthrough in Algorithm Design

Computer scientists at Carnegie Mellon University have devised an algorithm that might be able to solve a certain class of linear systems much more quickly than today's fastest solvers.

- As a result we're getting several inquiries
  - **"Do you have an implementation?"**

The **practice**:

our response to the inquiries

- “Not yet, but we do have a very fast solver”

$$2x - y = 5$$
$$5x + 2y = 8$$

$$2x - y = 5$$
$$\underline{-2x \qquad -2x}$$
$$-y = -2x + 5$$
$$y = 2x - 5$$

$$5x + 2(2x-5) = 8$$
$$5x + 4x - 10 = 8$$
$$9x - 10 = 8$$
$$\underline{+10 \quad +10}$$
$$9x = 18$$
$$\frac{9x}{9} = \frac{18}{9}$$
$$x = 2$$

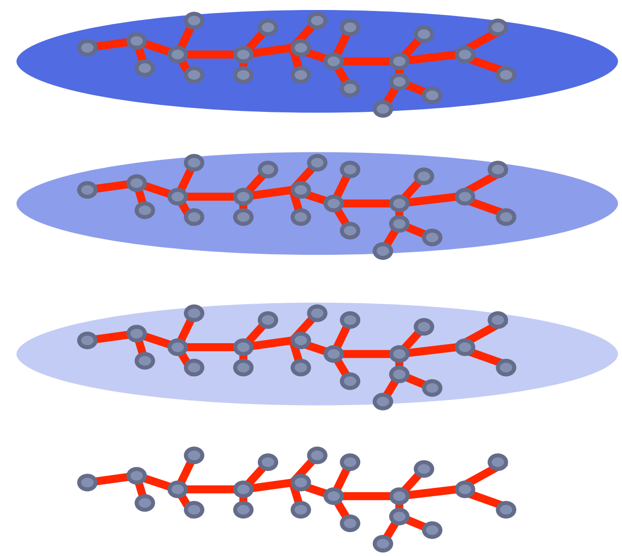
CMG: Combinatorial Multigrid

Combinatorial Multigrid is a **solver** for symmetric diagonally dominant linear systems.
CMG combines the strengths of **multigrid** with those of **combinatorial preconditioning**.

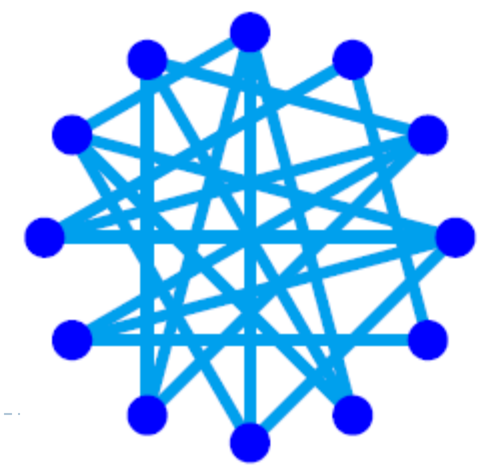
The solver runs in MATLAB. Download, take a look at readme, and install. Feel free to contact me.

Download

The **practice**:
The Combinatorial Multigrid Solver (CMG)



- In practice constants matter
- CMG has hard-to-beat constants and we understand why
- The chain makes information travel fast in the graph (rapid mixing)

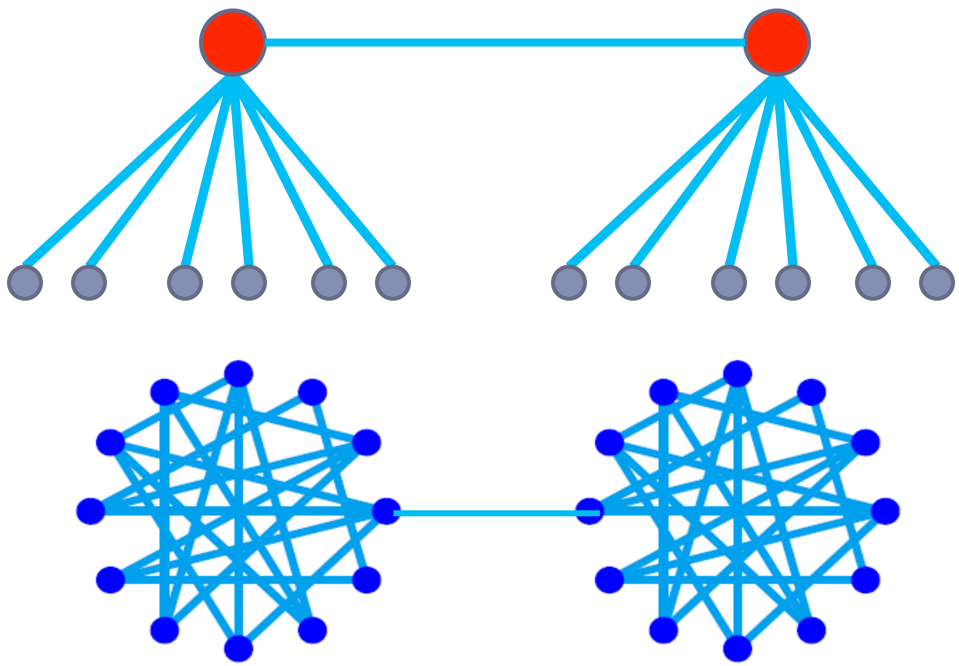


- Rapid mixing is inherent in well connected graphs

The **practice**:

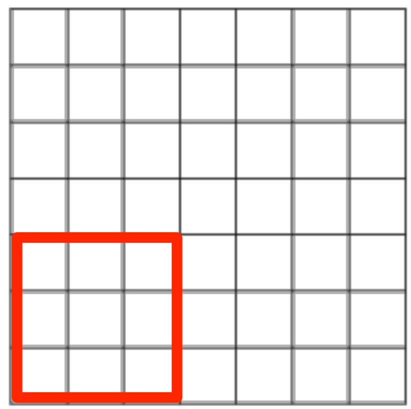
The Combinatorial Multigrid Solver (CMG)

- If it is possible to identify the **well-connected and isolated** components, we can actually construct a good preconditioner



The **practice**:
The Combinatorial Multigrid Solver (CMG)

- Every graph can be decomposed in good communities



- So, CMG is a **cut-based graph sparsification** algorithm
- Satisfactory decompositions in sparse graphs can be found quickly [Koutis, Miller 08]
- These give **better spectral approximations** for the same amount of size reduction, in all graphs with non-trivial connectivity

The **practice**:
The difficulties in CMG

- Satisfactory decompositions can be found in **sparse** graphs
- We do not have a practical and theoretically sound way of doing the same in **dense** graphs

- CMG constructs (recursively) a chain of graphs
- These can get dense so CMG may stagnate in recursion

- This is **rare**:
- Out of 3374 real-world graphs this appears in 26 cases [Livne, Brandt 12]

Bridging theory and practice:

Eventually Laplacian solvers will be a combination of cut-based **and** distance-based spectral sparsification

Bridging theory and practice:

full vs **incremental** spectral sparsification

- *Spielman & Srivastava:*
- Every graph has an **excellent** (fully) sparse spectral approximation
- It can be computed via sampling with probabilities proportional to effective resistances of the edges.

- An significant generalization was given in [Drineas, Mahoney 11]

- **Why incremental sparsification works:**
- Stretch is loose approximation to effective resistance
- We can compensate for "looseness" by extra sampling
- So instead of a **fully** sparse graph we get an **incremental** sparse one

Bridging theory and practice:
full spectral sparsification

- **Idea:** Whenever CMG yields a dense graph, sparsify it fully
- However we need to solve systems in order to compute them (sort of a chicken and egg problem)

Bridging theory and practice:

faster spectral sparsification [Koutis,Levin,Peng 12]

- Can we solve the chicken and the egg problem?

1. Use solvers but on **special graphs** on which they run faster
2. Compute the effective resistance over these special graphs
3. These will be somewhat crude approximations to the actual effective resistances (say by an $O(\log^2 n)$ factor)
4. Do **oversampling** and produce a slightly more dense graph ($O(n\log^3 n)$ edges)
5. Fully sparsify the last graph

Bridging theory and practice:

faster spectral sparsification [Koutis,Levin,Peng 12]

- Can we solve the chicken and the egg problem?

- Cheap solver-based solutions give:
  - An O(mlog n) time algorithm for graphs with $nlog^3\ n$ edges
  - An O(m) time **solver** for graphs with $nlog^5$ n edges

- This solution suggests the use of a distance-based solver as a subroutine to a cut-based solver (perhaps not so elegant)

Bridging theory and practice:
full spectral sparsification

- Can we **really** solve the chicken and the egg problem?

- Is there a **combinatorial** algorithm for spectral sparsification?
- It must work in time less than $O(m\log n)$

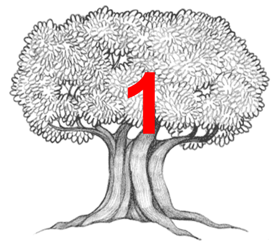
- The short answer is yes
- It is an iterative application of incremental sparsification

Graph **sketching** by trees

Theorem: Every graph can be spectrally approximated by a sum of $O(\log^3 n)$ trees.

Graph **sketching** by trees

log^2 n
trees



Remove
from graph

Remove
from graph

….

Keep the trees and apply
incremental sparsification
to the rest of the graph.

log^2 n
trees

Apply recursion….
log n levels each log^2 n trees

Conclusion

- *"A new breed of ultrafast computer algorithms offers computer scientists a novel tool to probe the structure of large networks." – from* ***"Network Solutions"***
- We've discovered practical algorithms
  - There is still room for improvement
- The combination of graph-theoretical algorithms and randomized linear algebra can produce extremely powerful algorithms
- The **hope** is that some of the methods will extend to more general linear systems

Thanks!