High Performance Linear System Solvers with Focus on Graph Laplacians

Richard Peng
Georgia Tech

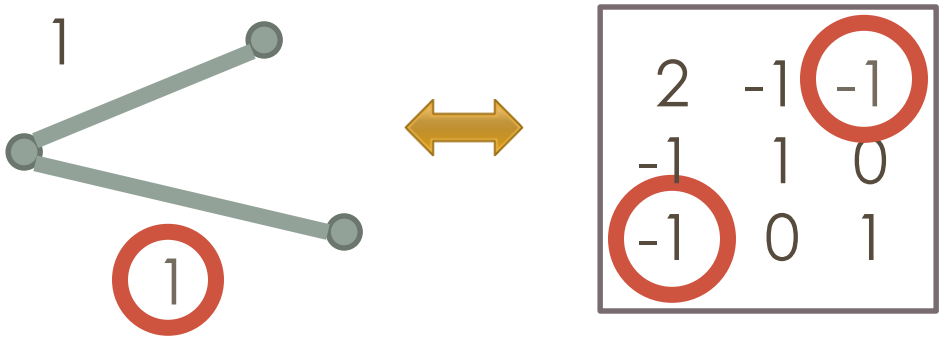
Co-PIs: John Gilbert (UCSB),
Gary Miller (CMU)

OUTLINE

- **Problem of Lx = b**
- Benchmarks and Evaluations
- Tree Based Solvers

GRAPH LAPLACIANS

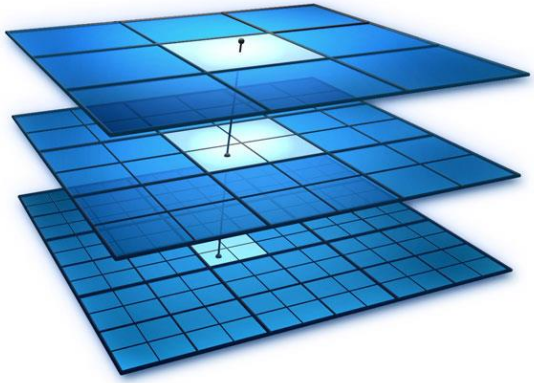
Matrices that correspond to undirected graphs

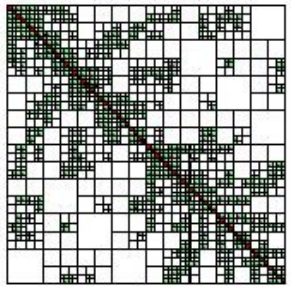


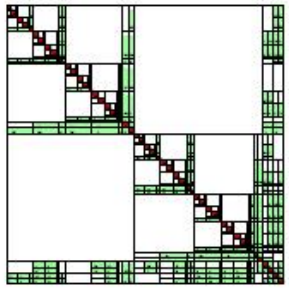
- Variables ⇔ vertices
- Non-zeros ⇔ edges

SOLVING **Lx = b**

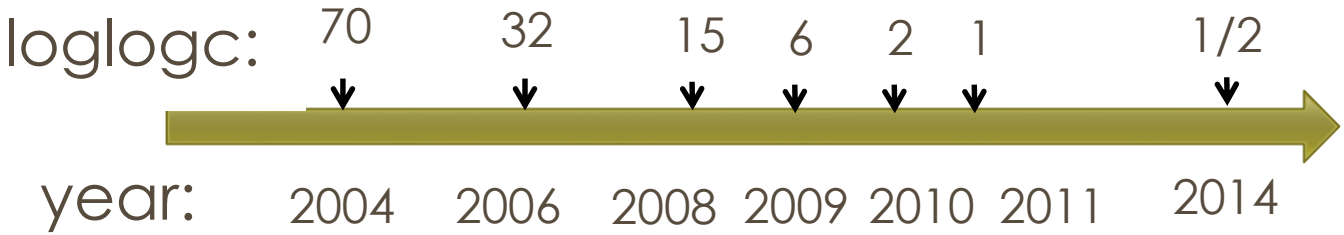
- Multigrid methods widely used in scientific computing
- Good runtimes for systems with as many as 10^9 nonzeros
- MATLAB: pcg(**L**, ichol(**L**), **b**, ε) 'works' for 10^6 nonzeros





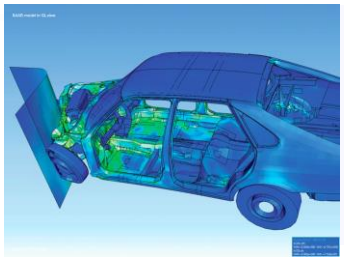


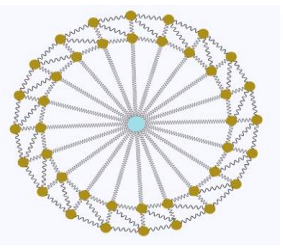
- [ST`04]: $O(m\log^c n \log(1/\varepsilon))$ time
- 2004 – 2014: c halved every 2 years



THE LAPLACIAN PARADIGM

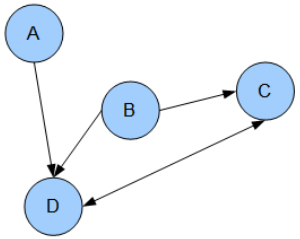
Directly related:
Elliptic systems

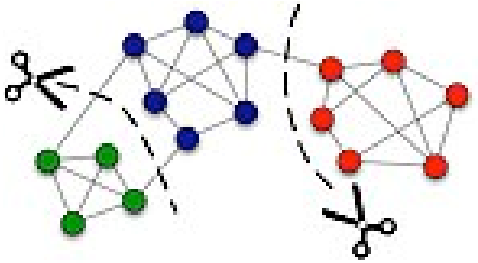




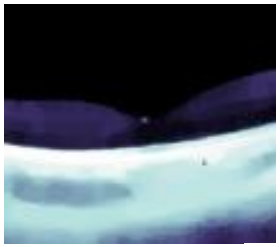
Lx=b

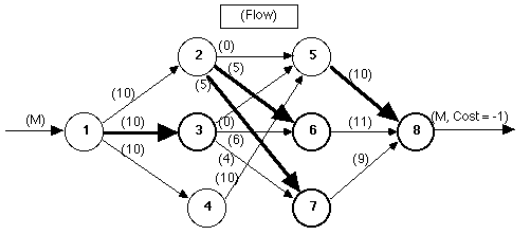
Few iterations:
Eigenvectors,
Heat kernels

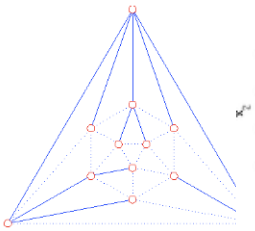


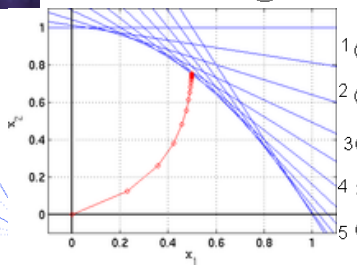


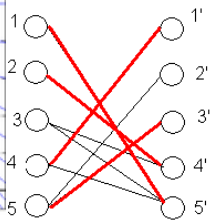
Many iterations / modify algorithm
Graph problems
Image processing



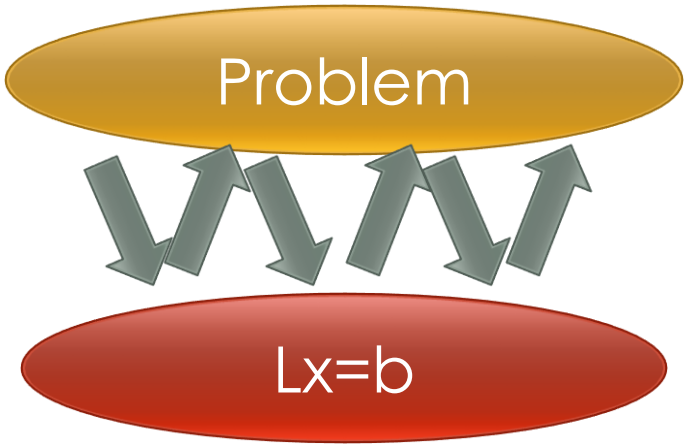








NEW WAYS OF USING SOLVERS



Sequence of (adaptively) generated linear systems:

- Power iteration
- Interior point method
- Iterative least squares

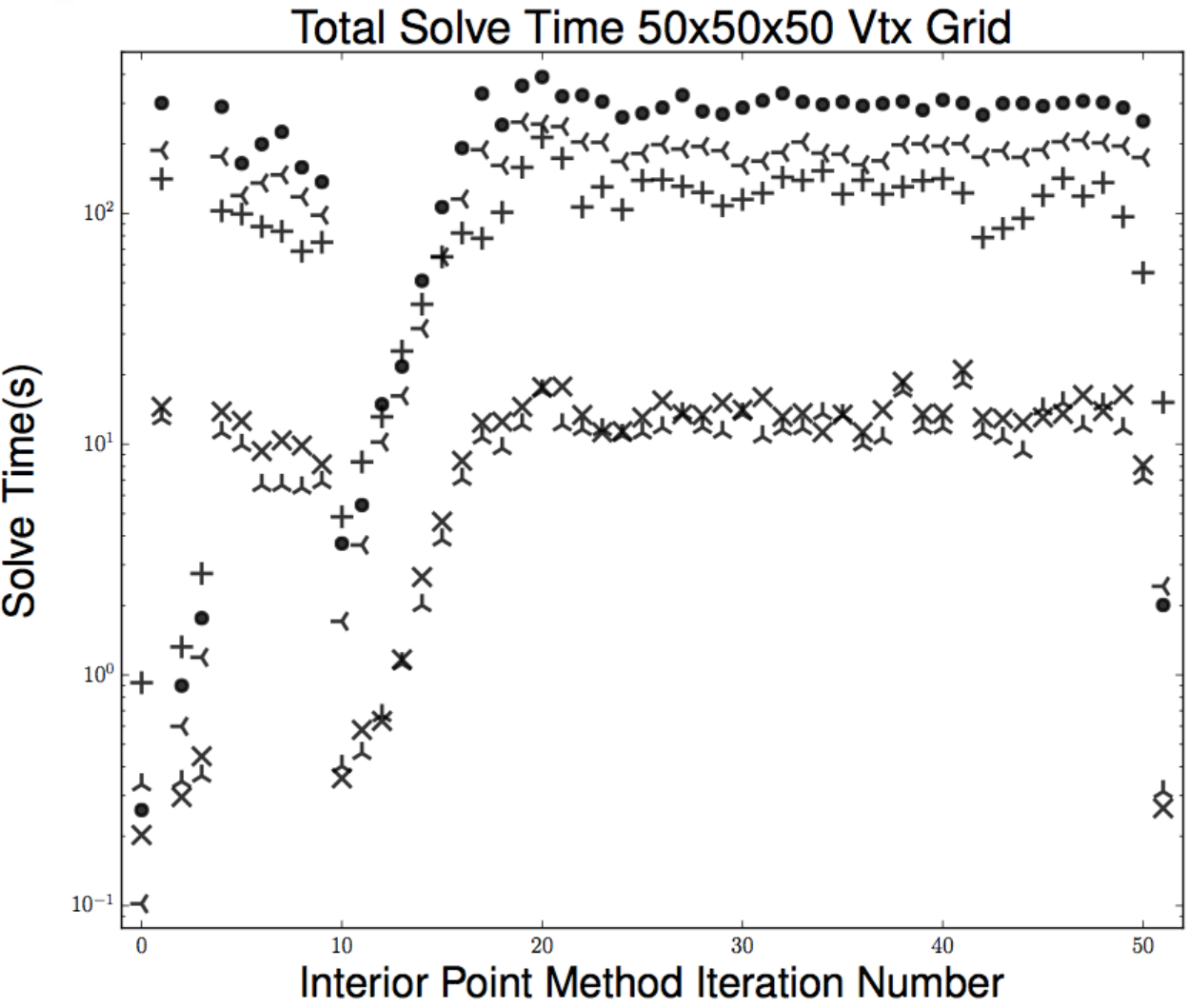
What makes such **L** and **b** hard:

- Widely varying weights
- Multiscale behavior
- Difficulties of the graph problems

OUTLINE

- Problem of Lx = b
- **Benchmarks and Evaluations**
- Tree Based Solvers

[KRS`15]: ISOTONIC REGRESSION



README file

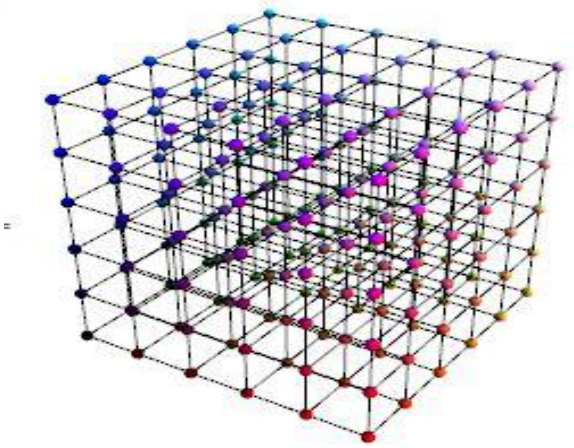
we suggest rerunning the program a few times and / or using a different solver. An alternate solver based on incomplete Cholesky is provided with the code.

https://github.com/sachdevasushant/Isotonic

GOAL: BENCHMARKS

Structured graphs

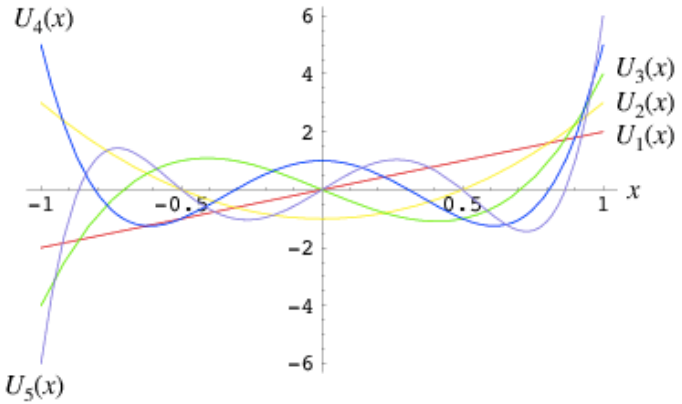
- Grids / cubes
- Cayley graphs
- Graph products



Hard graph problems

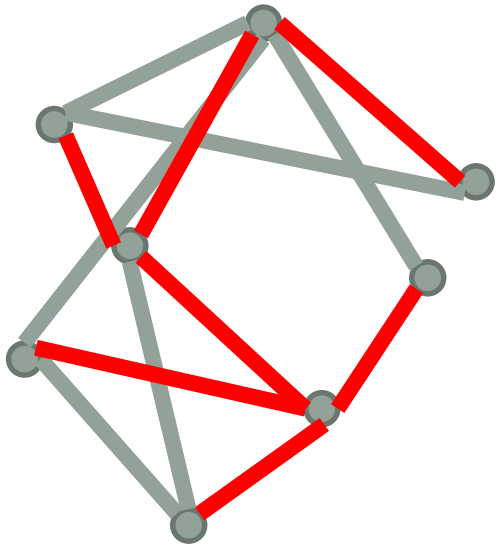
- Maxflow problems from DIMACS implementation challenges
- Linear systems arising from second-order optimization (IPM)

NUMERICS + COMBINATORICS:



Numerical methods (e.g. CG) rely on preconditioners

- Good approximation to **L**
- Easy (easier) to solve on



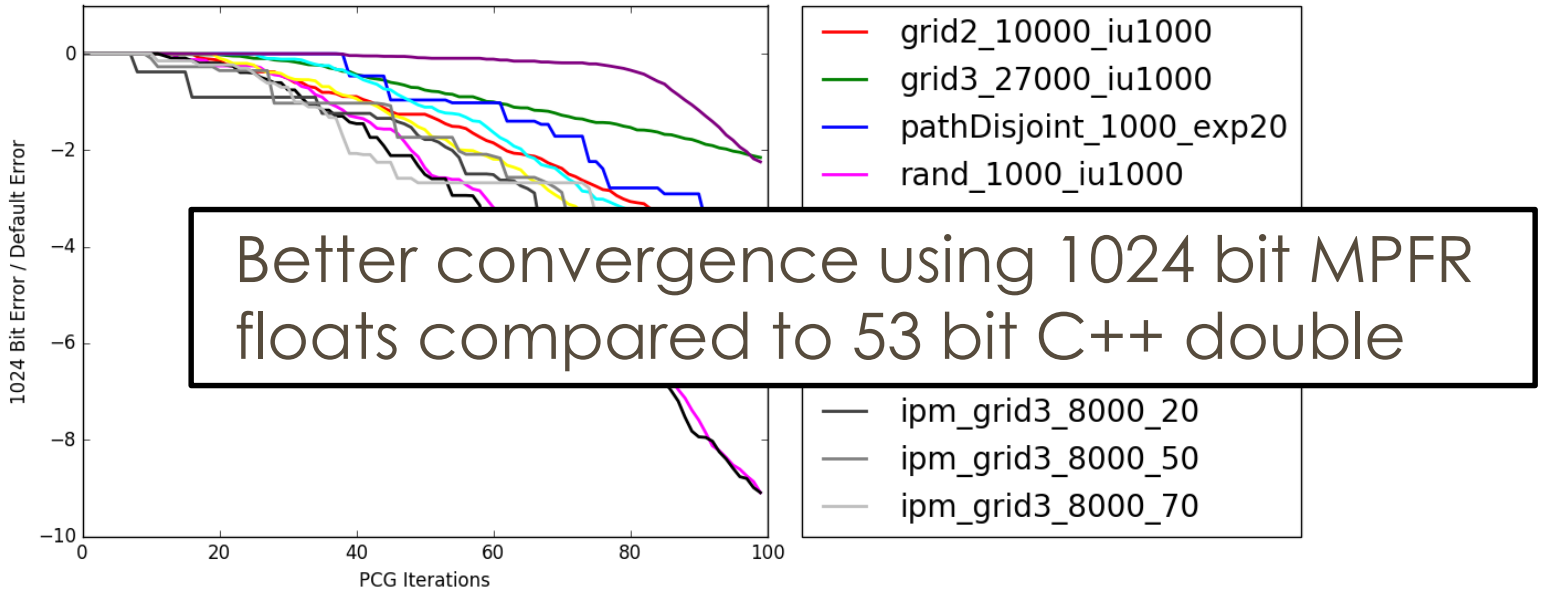
Spanning trees:

- finite approximation
- linear time solve

NUMERICS + COMBINATORICS

Conjugate gradient (CG) with tree preconditioner:

- [textbook]: $m^{1/2}$ iters, even with round-off errors
- [SW`09]: with exact arithmetic, takes $m^{1/3}$ iters

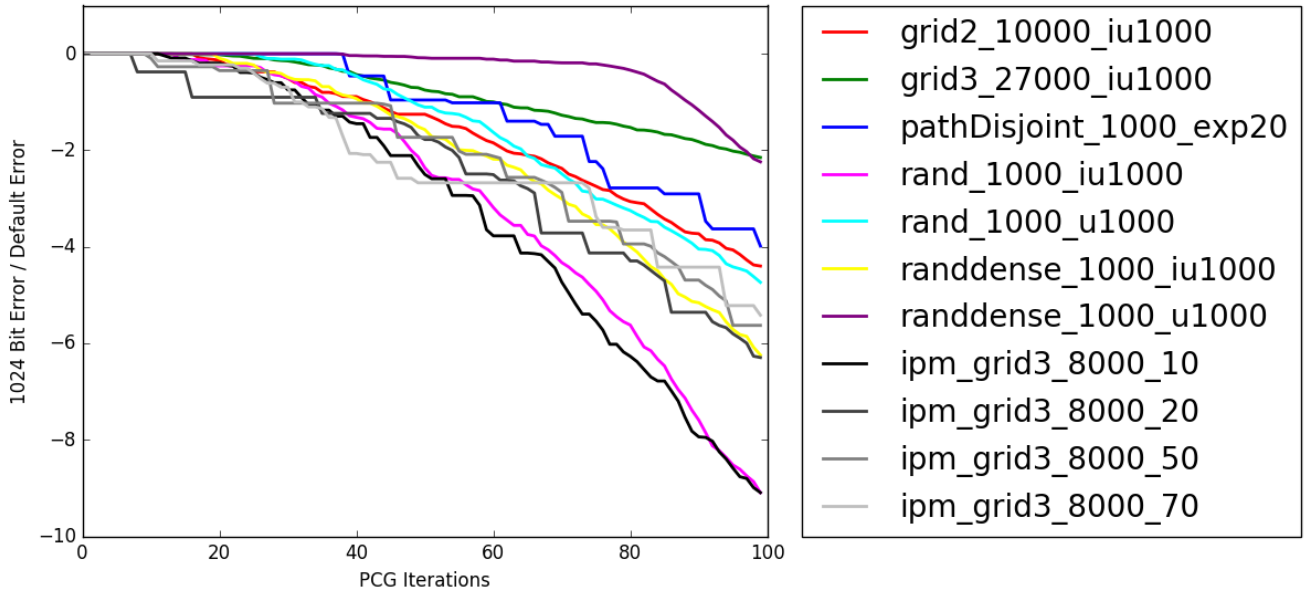


Better convergence using 1024 bit MPFR floats compared to 53 bit C++ double

https://github.com/serbanstan/TreePCG
https://github.com/danspielman/Laplacians.jl

QUESTION: NUMERICAL PRECISION

- Can numerical precision be analyzed through the graph theoretic components?
- Primal-dual view of precision? CG?



https://github.com/serbanstan/TreePCG
https://github.com/danspielman/Laplacians.jl

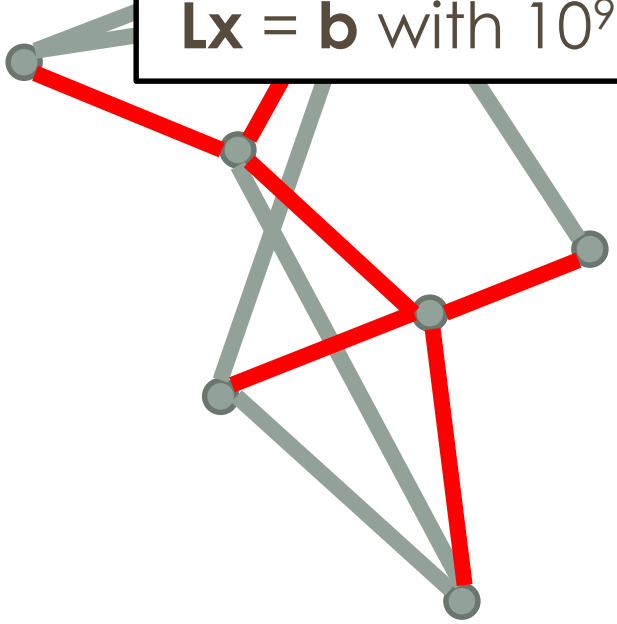
OUTLINE

- Problem of Lx = b
- Benchmarks and Evaluations
- **Tree Based Solvers**

GOAL: FAST TREE-BASED SOLVERS

Gradually transform a tree-based solution to a solution on the entire graph

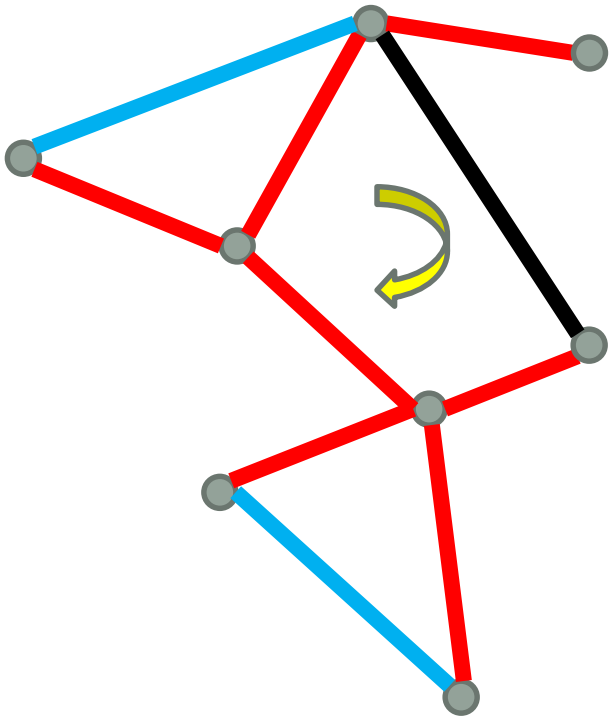
Claim: these ideas lead to code that can solve **any** $\mathbf{Lx} = \mathbf{b}$ with 10^9 edges in ≤ 10 seconds on ≤ 64 cores



Method	Cycle Toggle	Ultrasparsifier
Cost / Iter	logn	$m + (m/k)^2$
# Iters	$m\log^{1/2}n\log(1/\varepsilon)$	$k^{1/2}\log(1/\varepsilon)$
Related to	SGD	Grad. descent
Step uses	Data structures	Mat-Vec multiply

CYCLE TOGGLING

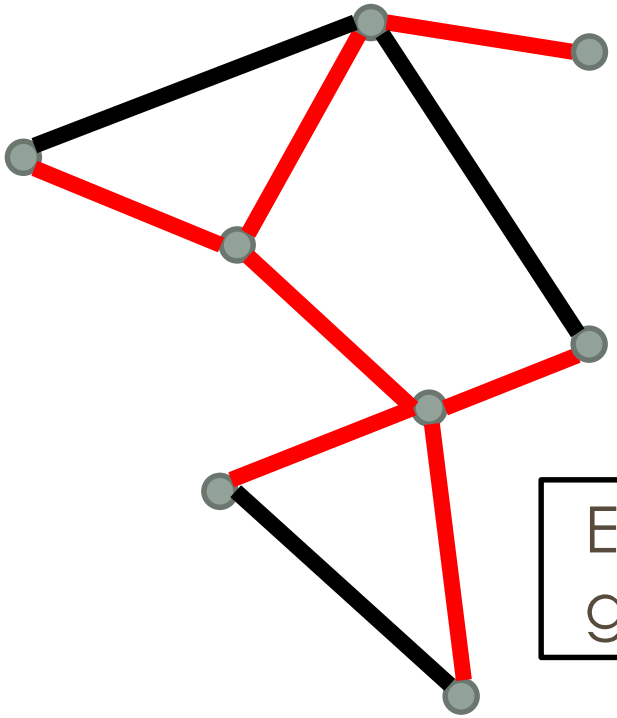
- Pick one off tree edge e at a time, make progress using T + e as preconditioner
- Speed up calculations using data structures



- [KOSZ `13]: akin to toggling dual flow along cycle, mlogn toggles, each costing O(logn)
- [LS `13]: CG-like acceleration to $O(m\log^{1/2}n)$ toggles

AUGMENTED TREES

- Add some edges to a tree to form a `batched' preconditioner
- Use exact methods on preconditioner

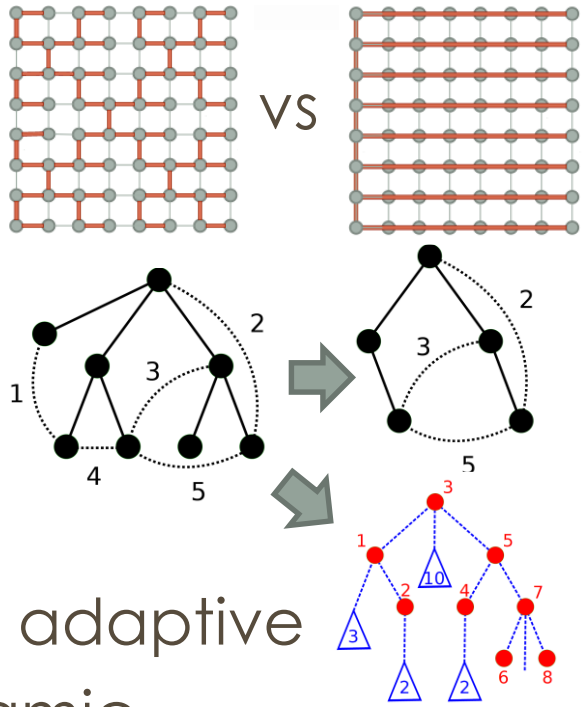


- [Vaidya `91]: MST + edges
- [KMP`10]: $O(m\log^2 n/k)$ edges ➔ $k^{1/2}$ iters
- Optimize: $m^{5/4}\log^{1/2} n$

Exists recursive versions, but those gains only kick in at around 10^9 edges

MOVING PIECES

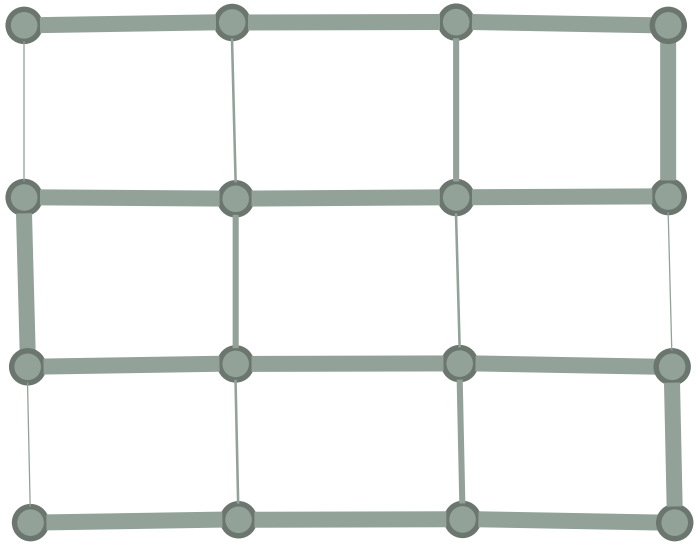
Method	Cycle Toggle	Ultrasparsifier
Cost / Iter	logn	$m + (m/k)^2$
# Iters	$m\log^{1/2}n\log(1/\varepsilon)$	$k^{1/2}\log(1/\varepsilon)$
Related to	SGD	Grad. descent
Step uses	Data structures	Mat-Vec multiply

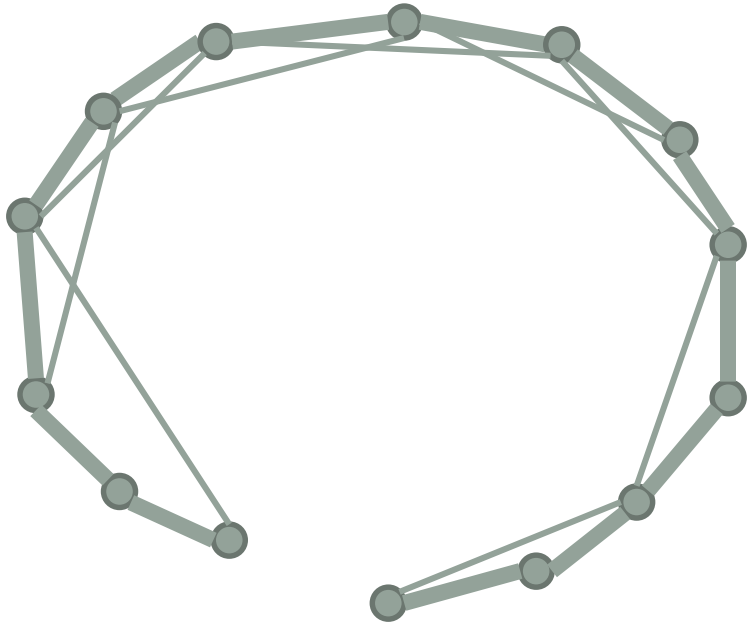


- Trees: MST / bottom-up / top-down / adaptive
- Data structures: offline / static / dynamic
- Numerics: batched / local, accelerated / CG
- Initialization: tree solution / recursive

BENCHMARK FOR TREE BASED ALGOS: HEAVY PATH GRAPHS

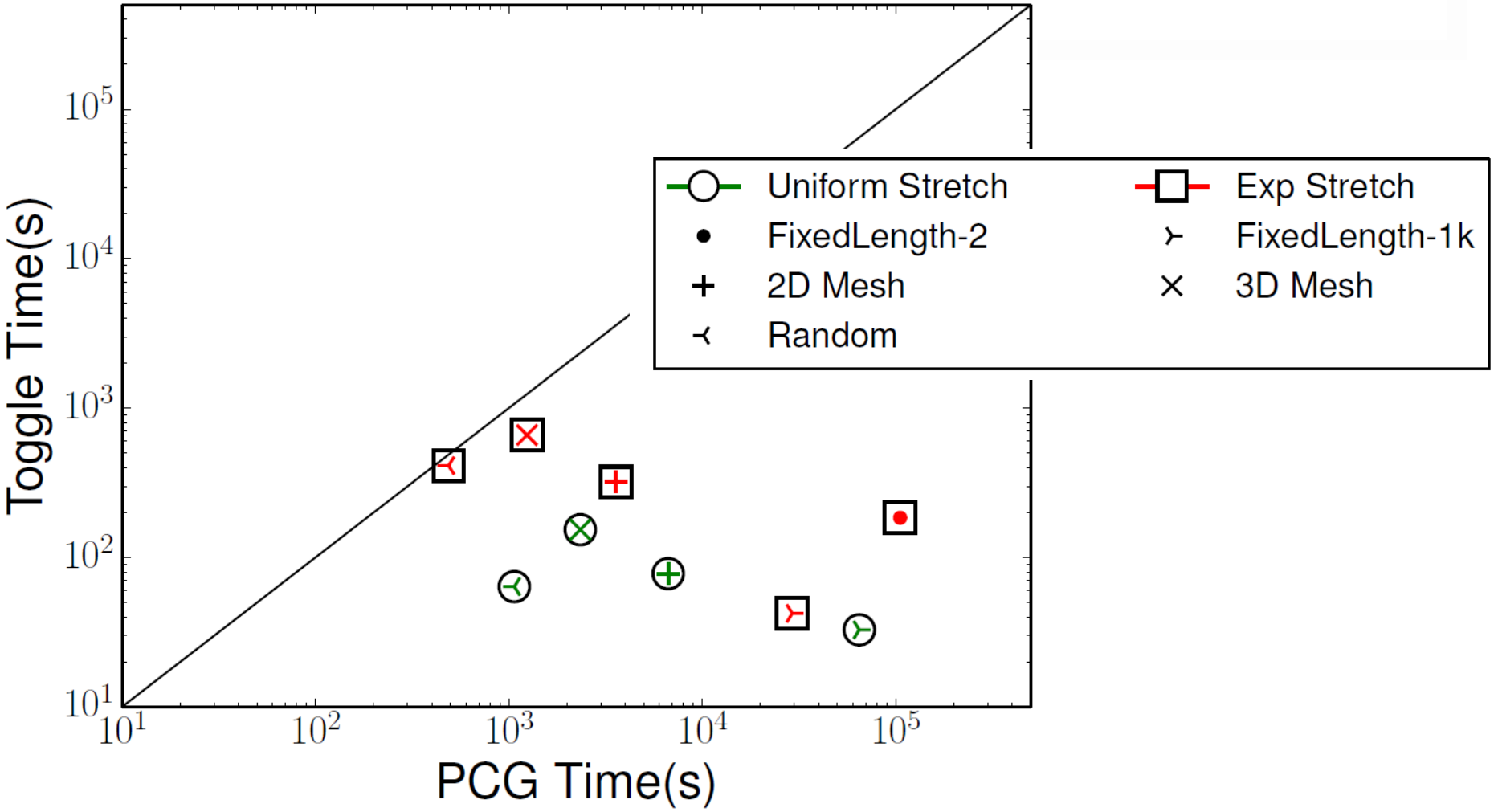
Pick a Hamiltonian path, weight all other edges so each has stretch 1





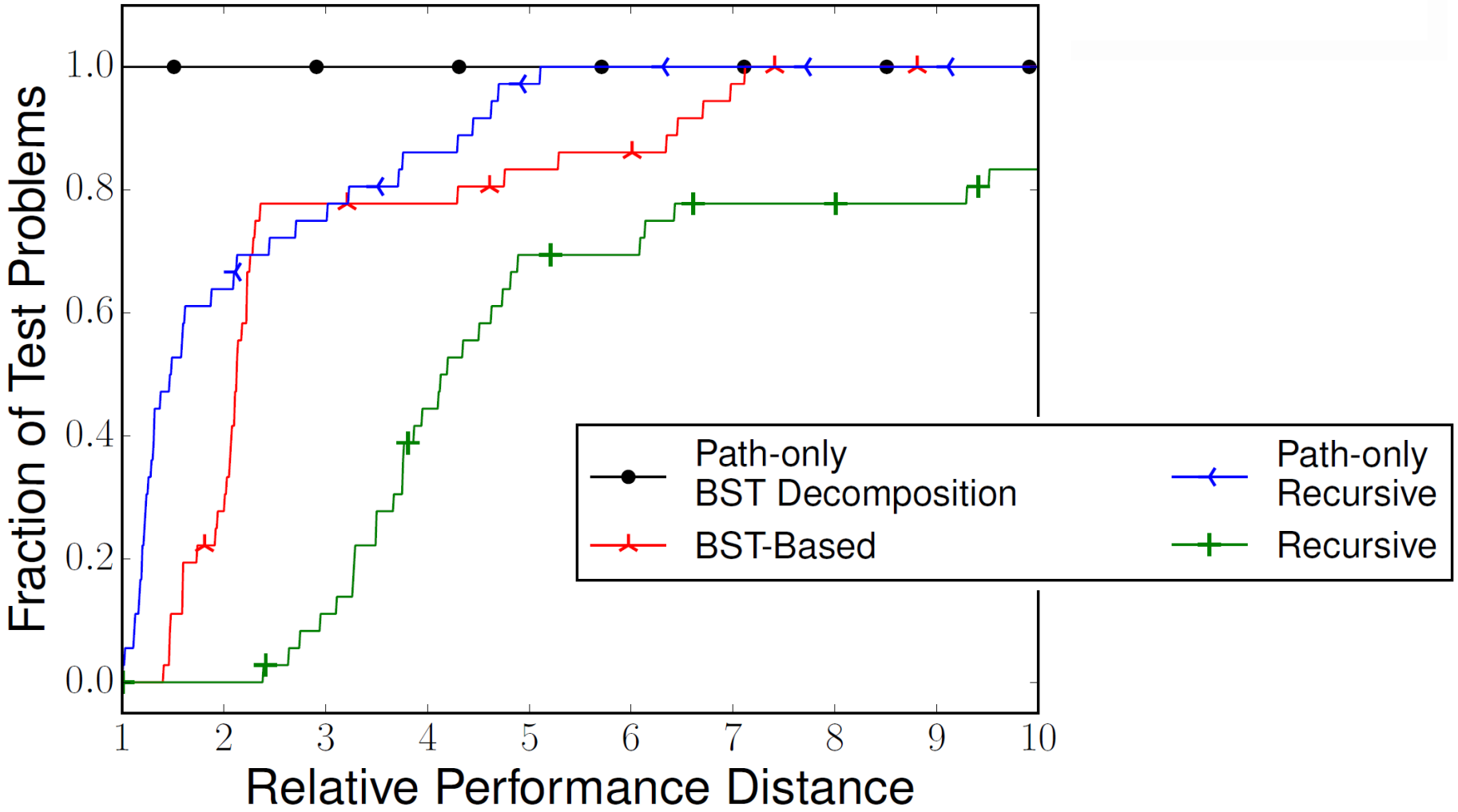
- Bad case for PCG,
- `easy' for tree data structures

CYCLE TOGGLING VS. PCG



https://arxiv.org/abs/1609.02957
https://github.com/sxu/cycleToggling

VARIANTS OF CYCLE TOGGLING



https://arxiv.org/abs/1609.02957
https://github.com/sxu/cycleToggling

THANK YOU

- Collaborators:
  - Hui Han Chin (CMU),
  - Kevin Deweese (UCSB),
  - John Gilbert (UCSB),
  - Gary Miller (CMU),
  - Saurabh Sawlani (GaTech),
  - Serban Stan (Yale),
  - Haoran Xu (MIT),
  - Shen Chen Xu (CMU)
- Repos & Papers:
  - https://github.com/sxu/cycleToggling
  - https://github.com/serbanstan/TreePCG
  - https://github.com/danspielman/Laplacians.jl
  - https://arxiv.org/abs/1609.02957