High Performance Linear System Solvers with Focus on Graph Laplacians

Richard Peng
Georgia Tech

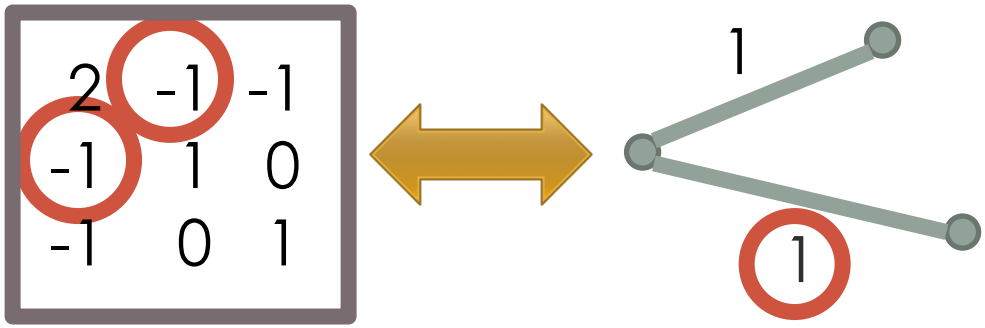
Based on work joint with: Serban Stan, Shen Chen Xu, Saurabh Sawlani, John Gilbert, Kevin Deweese, Gary Miller, Hui Han Chin

OUTLINE

- **Laplacian solvers and applications**
- Combinatorial preconditioning
- Numerics of tree preconditioners

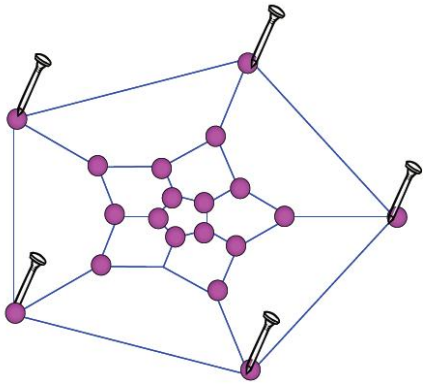
GRAPH LAPLACIAN MATRIX

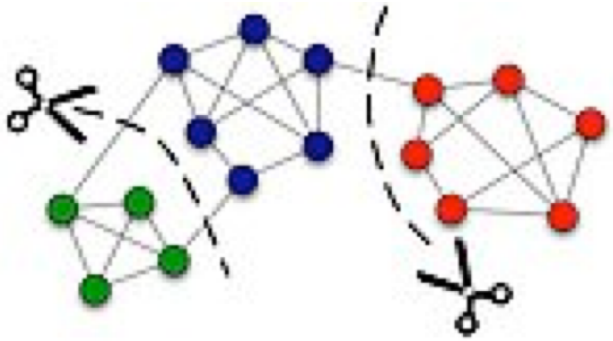
- Diagonal: weighted degrees
- Off-diagonal: -edge weights

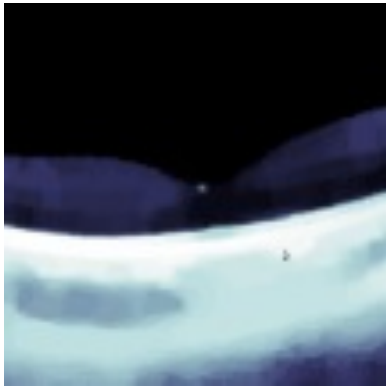


FEW ITERATIONS OF $\mathbf{Lx = b}$

- [Tutte `61]: graph drawing,
- [ZGL `03], [ZHS `05] [KRS `15]: learning/inference





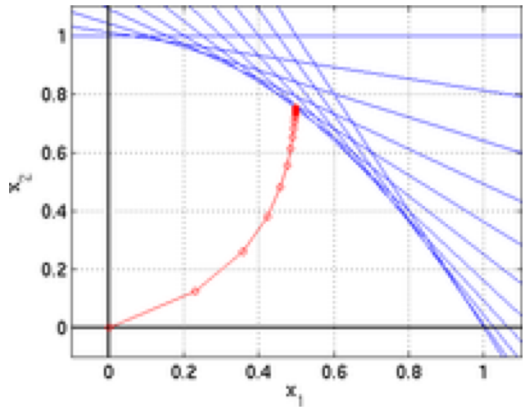


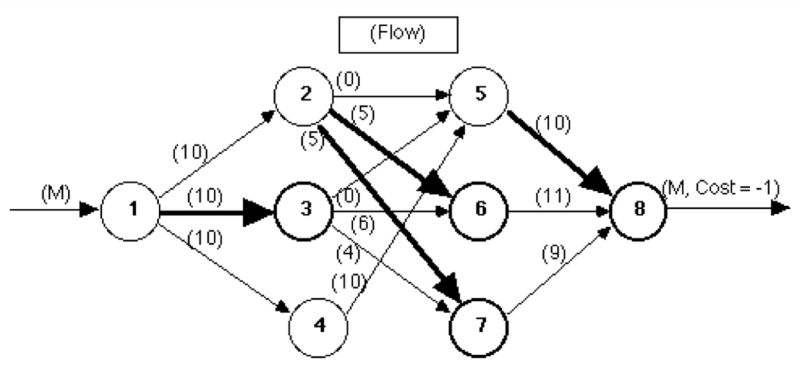
Inverse powering: eigenvectors / heat kernel:

- [AM `85] [OSV `12]: clustering
- [SM `01][KMST `09]: image segmentation

MANY ITERATIONS OF $\mathbf{Lx = b}$

[Karmarkar, Ye, Renegar, Nesterov, Nemirovski …]: convex optimization via. solving $O(m^{1/2})$ linear systems





[DS `08][CKMST `11][LS `14][AKPS `19][APS`19][AS `20]: graph problems → Laplacian linear systems

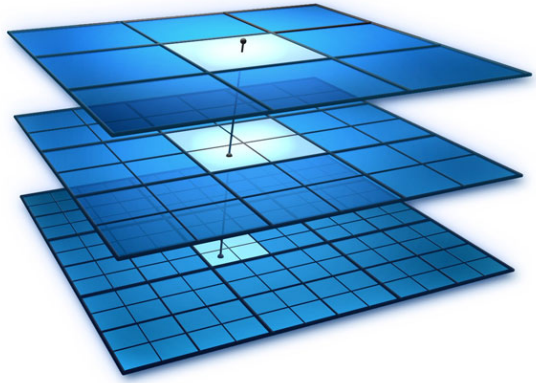
LINEAR SYSTEMS SOLVERS

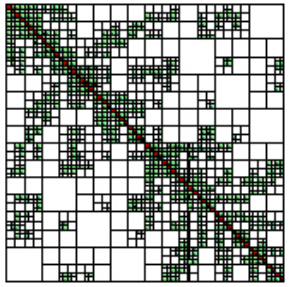
General systems:

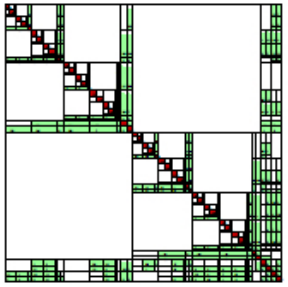
- Matrix multiplication: $O(n^{2.372864\ldots})$
- Conjugate gradient: $O(nnz\ k^{1/2})$ where k is condition number
- In practice: Jacobi iteration, Gauss-Siedel iteration, multigrid

$\mathbf{Lx} = \mathbf{b}$ in Practice

- 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

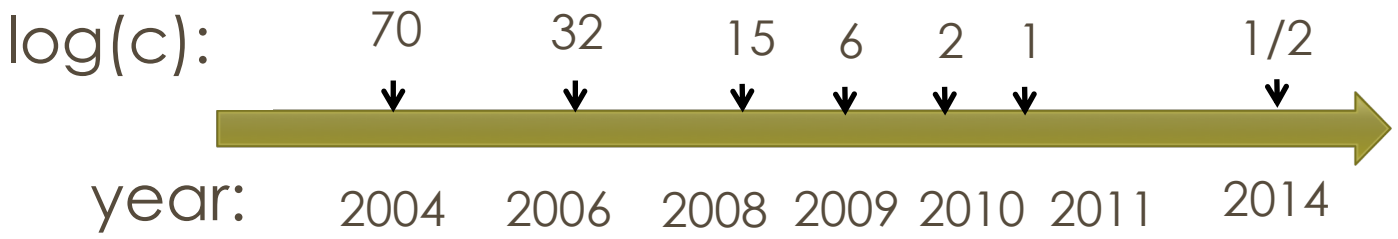






COMBINATORIAL PRECONDITOINING

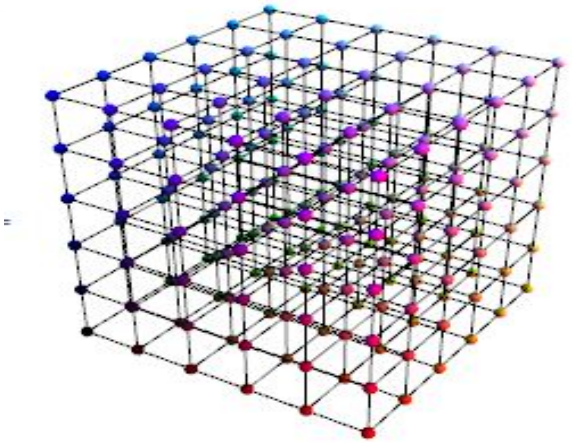
- [Vaidya `89]: use graph theory to build preconditioners for L
- [ST`04]: $O(m\log^c n\log(1/\varepsilon))$ time
- 2004 – 2014: c halved every 2 years



COMPARE? NEW BENCHMARKS:

Structured graphs

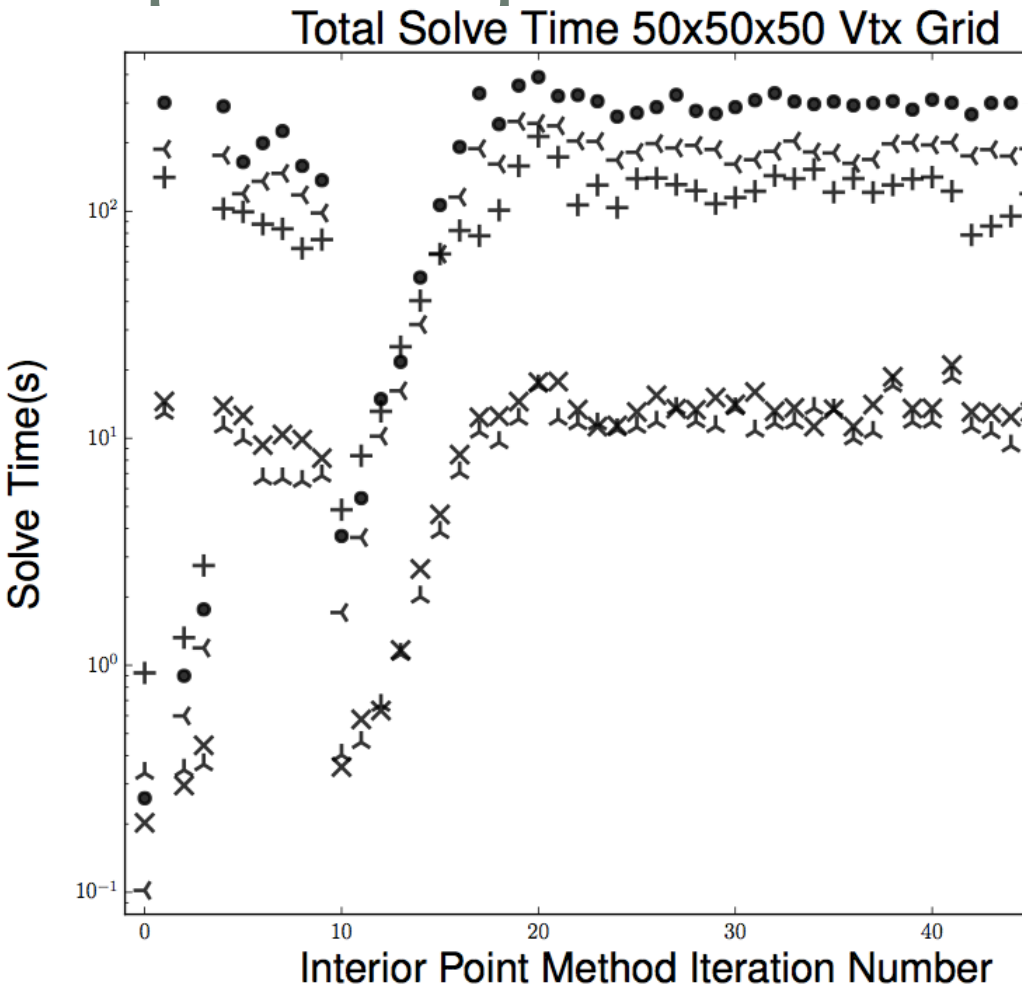
- Grids / cubes
- Cayley graphs
- Graph products



Hard graph problems

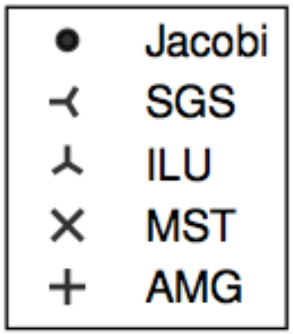
- Maxflow problems from DIMACS implementation challenges
- Linear systems arising from second-order optimization (IPM)

[KRS`15] + DIFFERENT SOLVERS



README file at https://github.com/sachdevasushant/Isotonic

we suggest rerunning the program a few times and / or using a different solver.



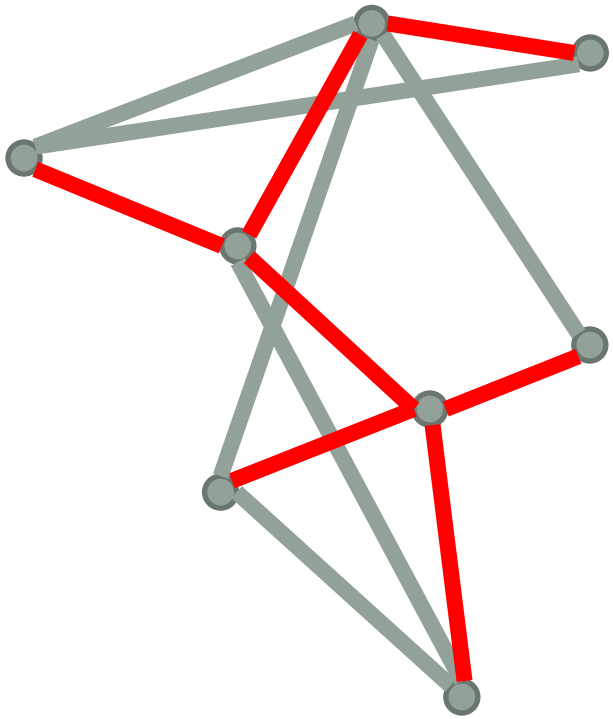
Disclaimer: this behavior is also depend heavily on numerics / termination conditions

OUTLINE

- Laplacian solvers and applications
- **Combinatorial preconditioning**
- Numerics of tree preconditioners

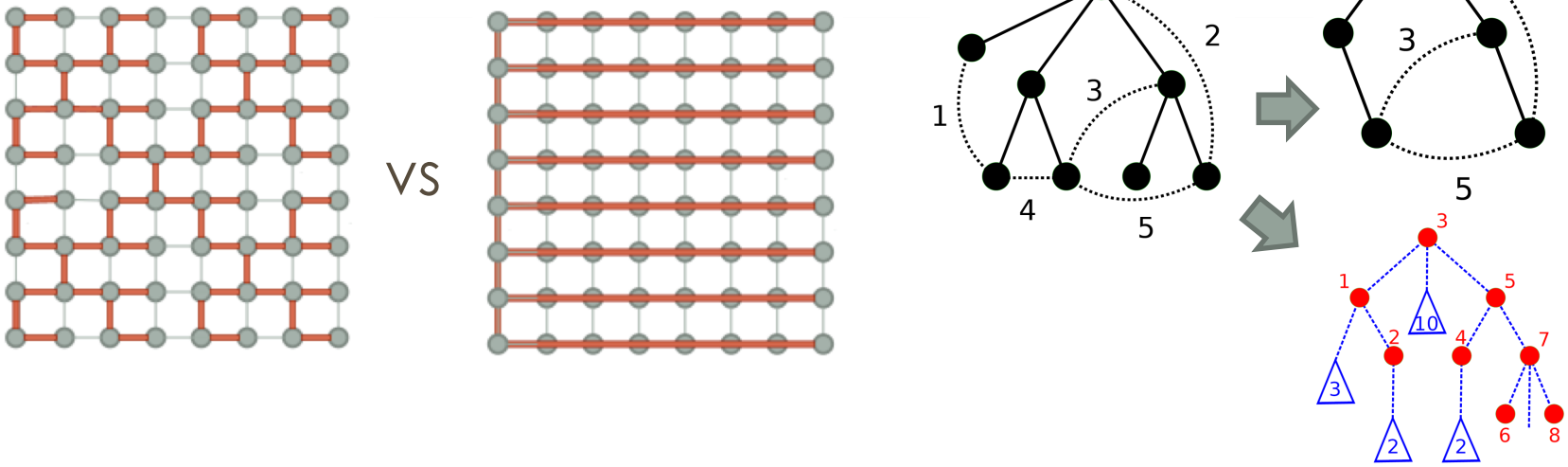
TREE BASED PRECONDITIONERS

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



Method	Cycle Toggle	Precondition
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
Primitives	Data structures	Mat-Vec

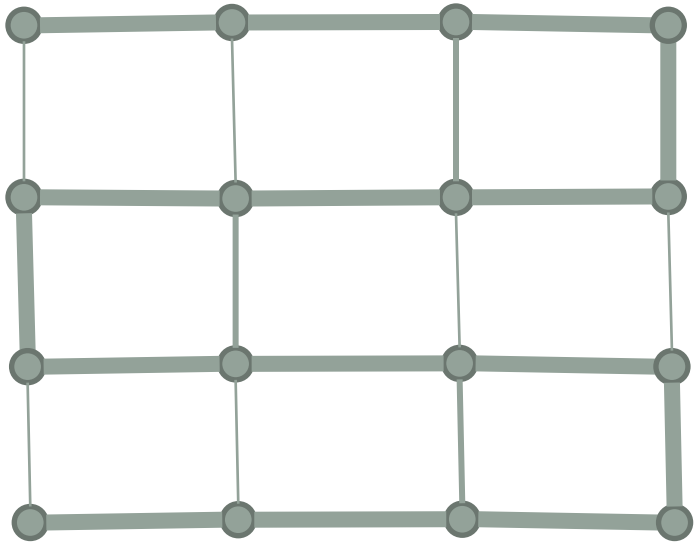
MOVING PIECES

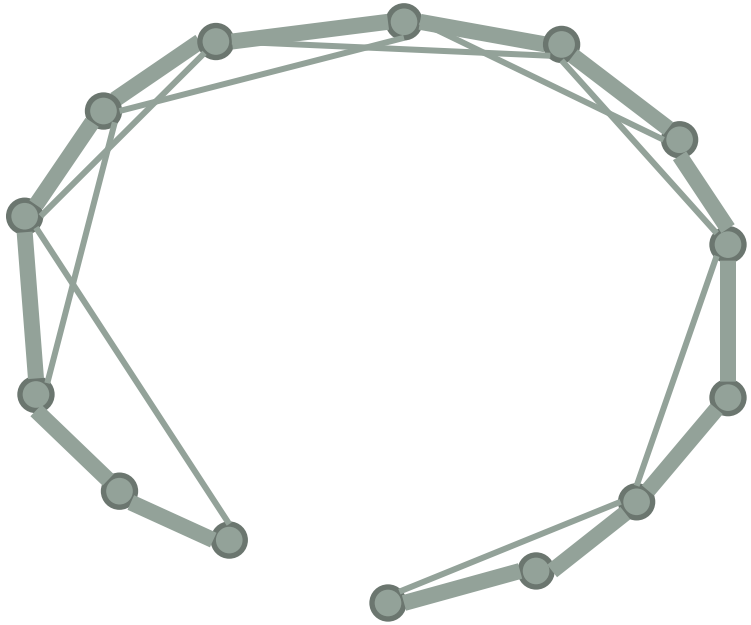


- Trees: MST / bottom-up / top-down / adaptive
- Data structures: offline / static / dynamic
- Numerics: batched / local, accelerated / CG
- Initialization: tree solution / recursive

[CSC `16] BENCHMARK FOR TREE BASED ALGOS: HEAVY PATH GRAPHS

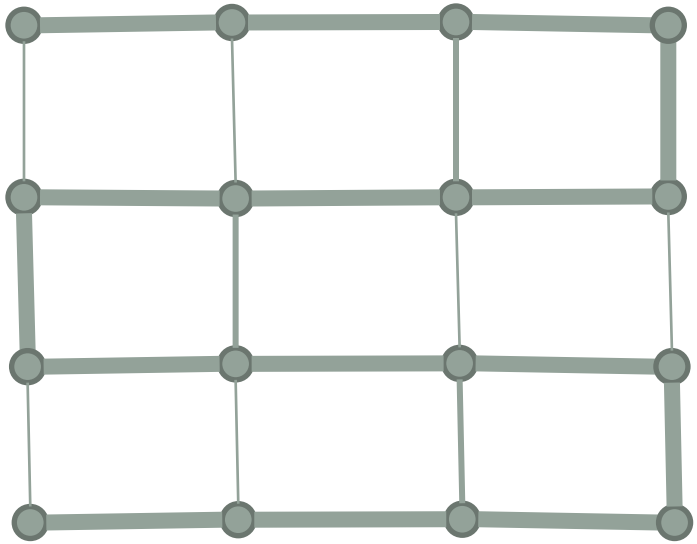
Pick a Hamiltonian path, weight all other edges so each has stretch 1

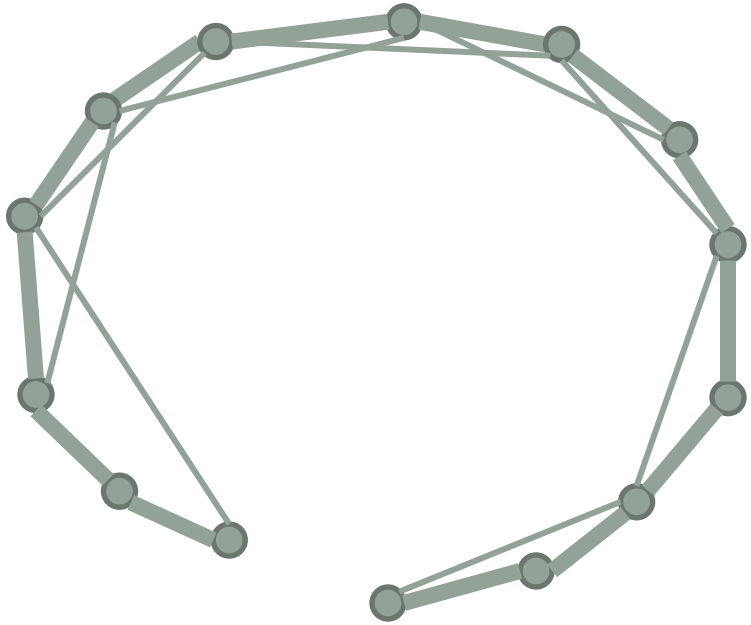




[CSC `16] BENCHMARK FOR TREE BASED ALGOS: HEAVY PATH GRAPHS

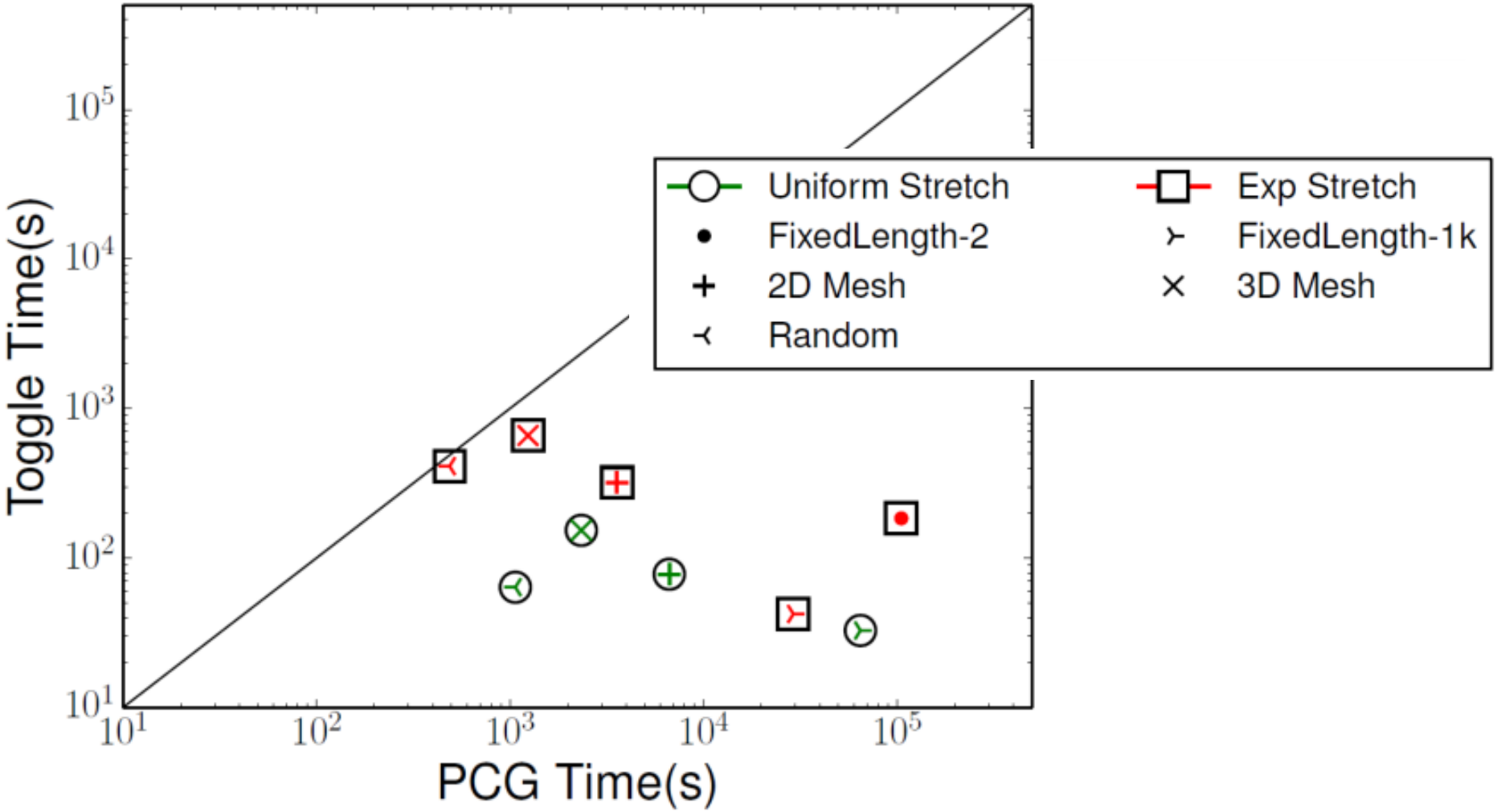
Pick a Hamiltonian path, weight all other edges so each has stretch 1





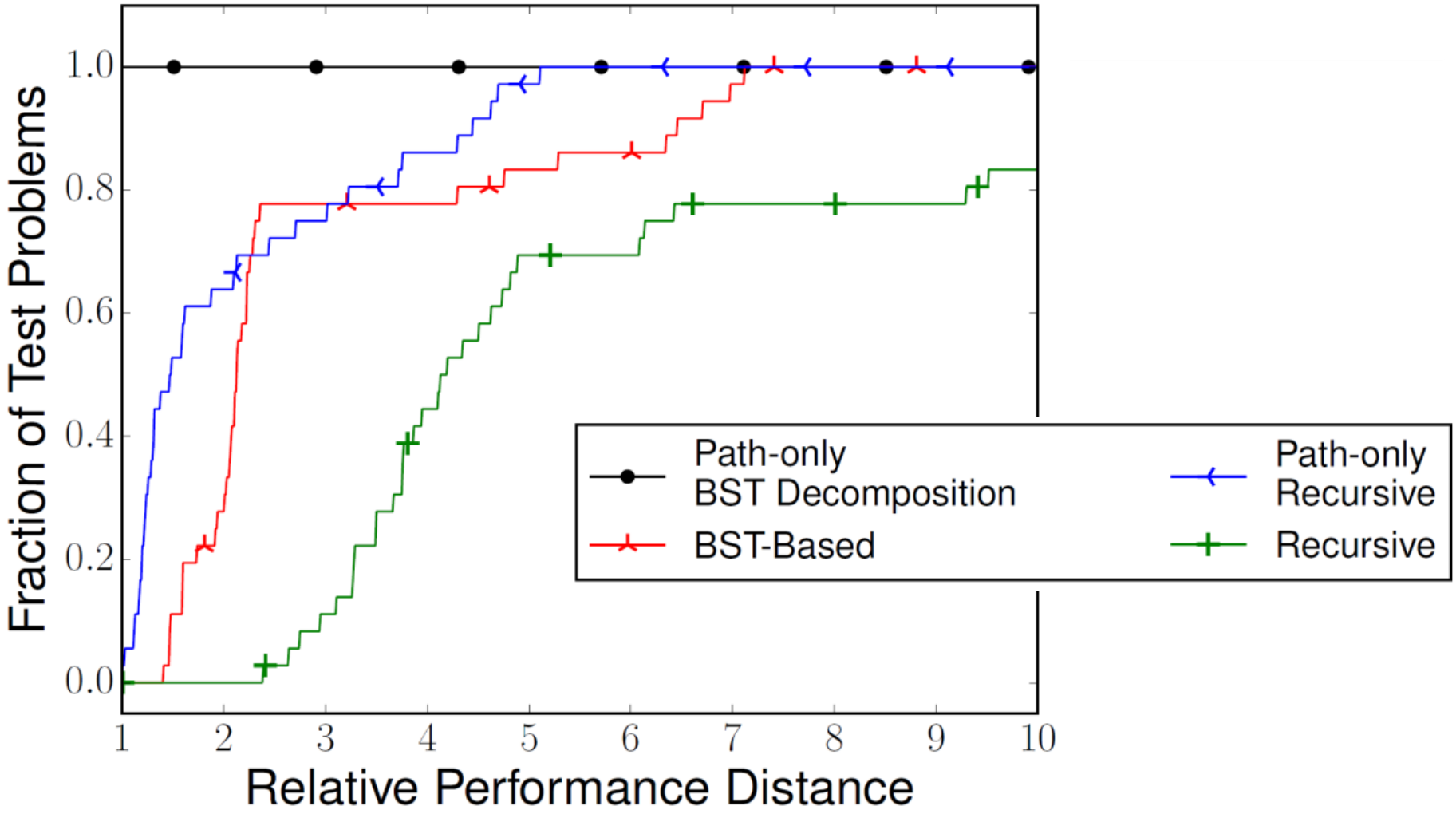
- Bad case for PCG,
- `easy' for tree data structures

DOING BETTER THAN CG



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

HOW TO GO FROM GRAPHS TO TREES?

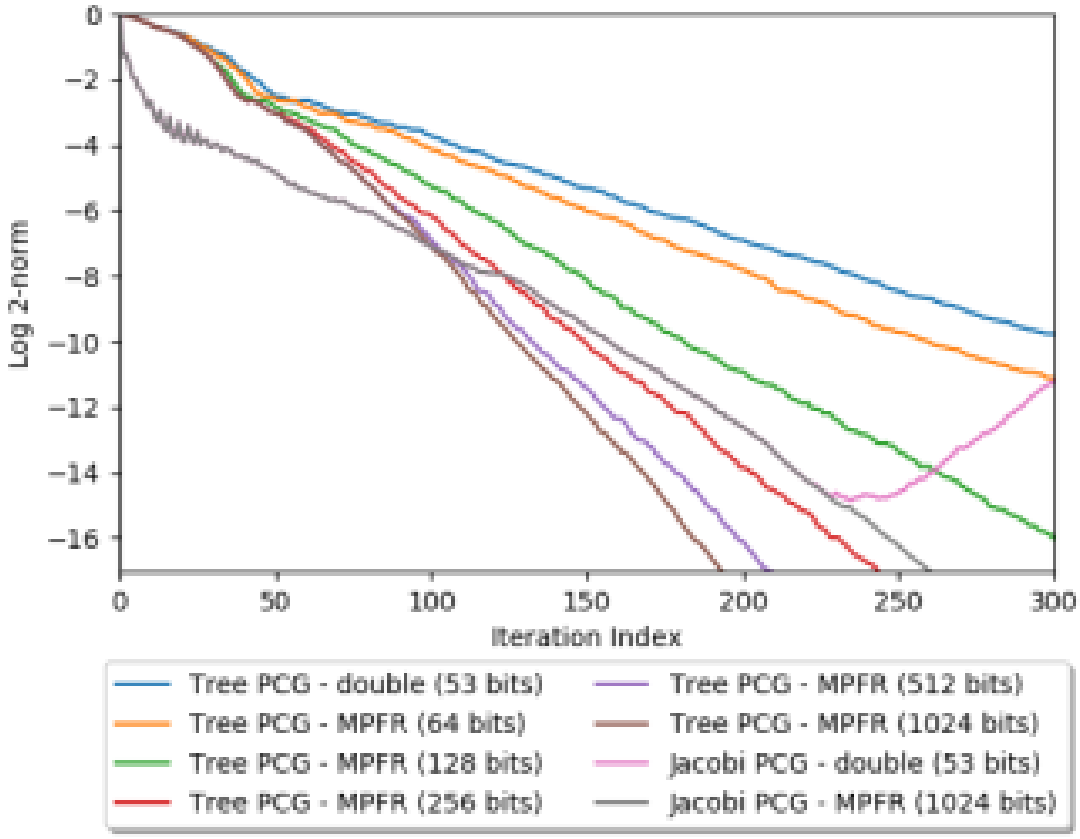


https://arxiv.org/abs/1609.02957
https://github.com/sxu/cycleToggling

OUTLINE

- Laplacian solvers and applications
- Combinatorial preconditioning
- **Numerics of tree preconditioners**

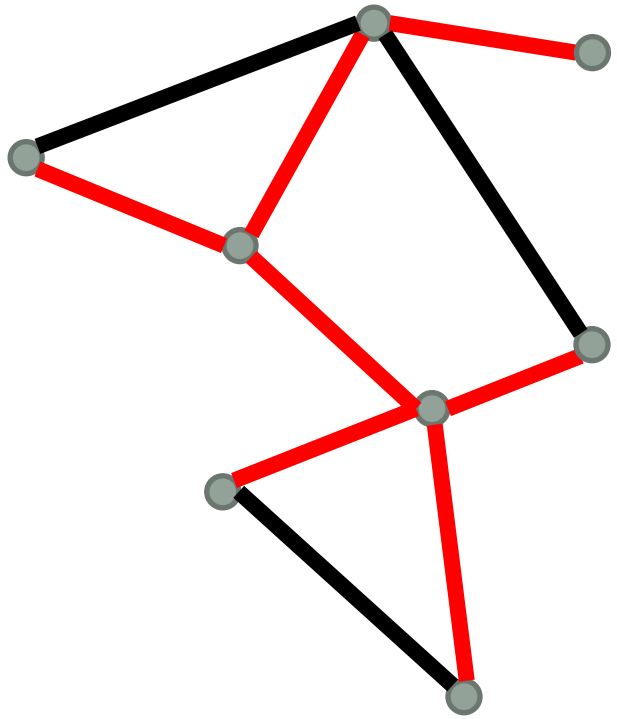
[CSC `20] NUMERICAL DIFFICULTIES OF COMBINATORIAL PRECONDITIONING



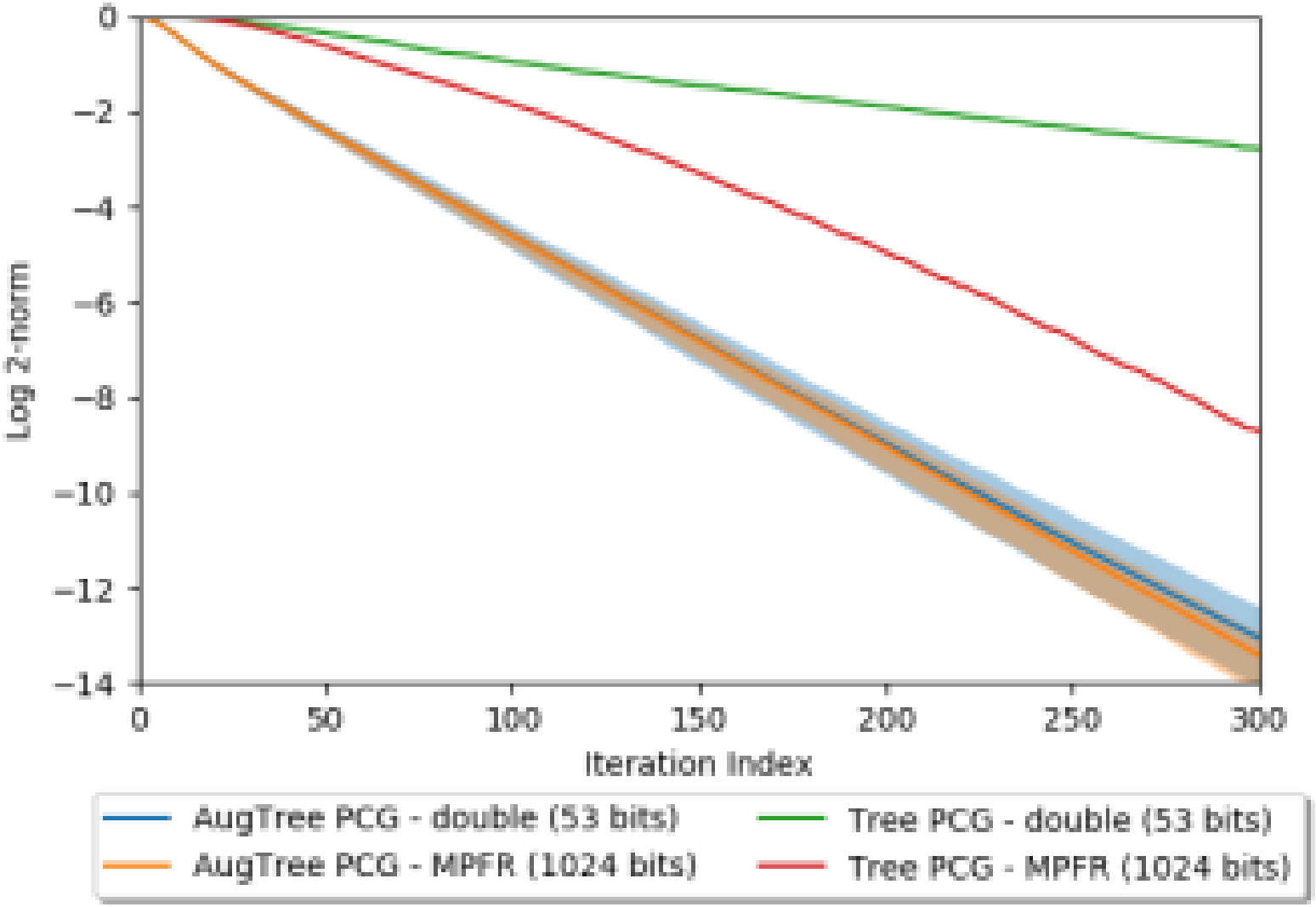
CG with a low-stretch tree as preconditioner

ONE FIX: BATCHED PROCESSING

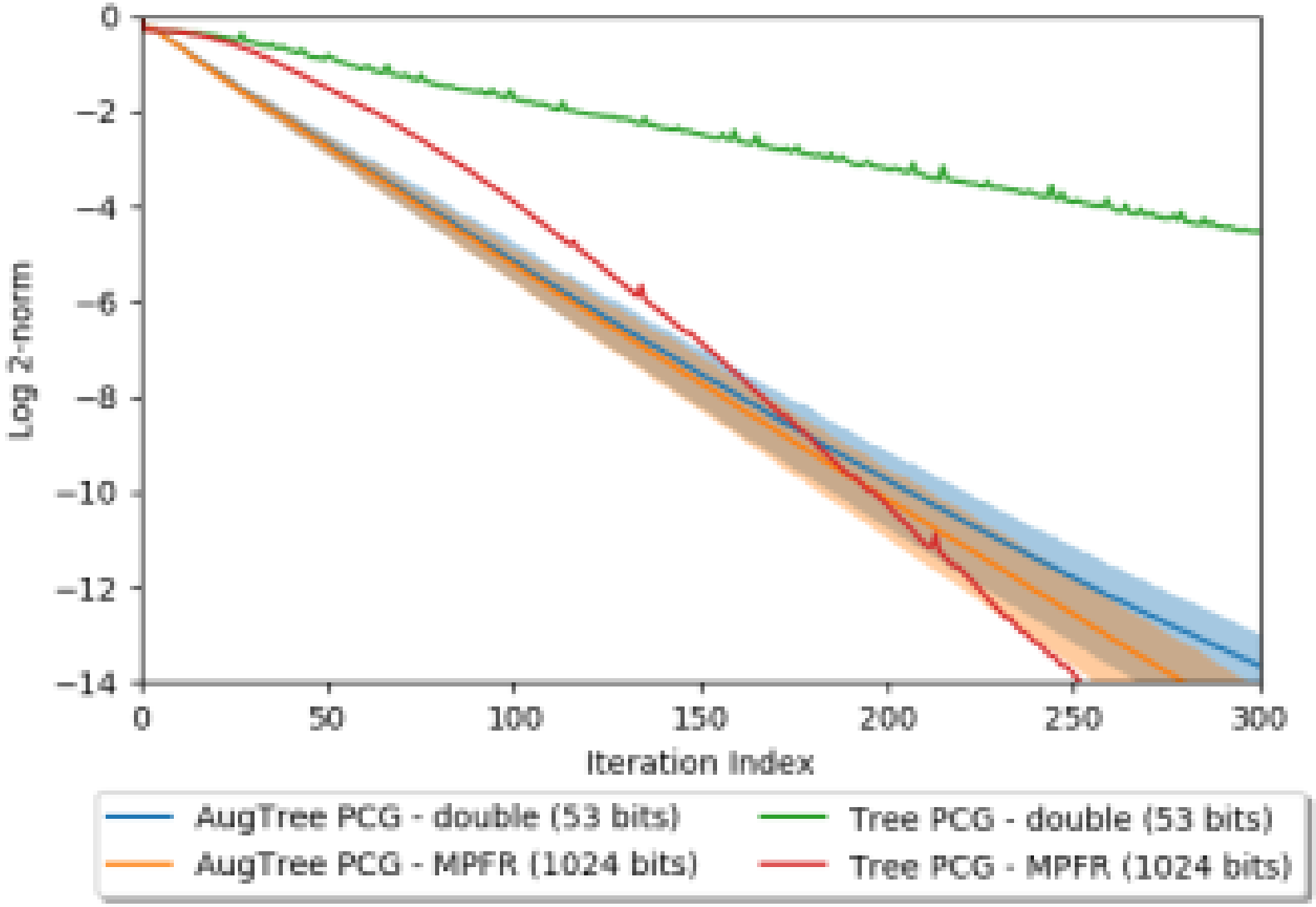
- Add some edges to a tree to form a `batched' preconditioner
- Use direct methods to factorize preconditioner explicitly



COMPARISON ON 3D CUBE



COMPARISON ON IPM MATRIX



EVALUATING LAPLACIAN SOLVERS

- We have a much better idea of what are the instances to test on now:
  - Weighted grid graphs
  - Inner loops of optimization algorithms
  - [Deweese-Gilbert `18]: evolve them!
- Measuring numerical behaviors?
- Benchmark w.r.t. applications: ground-truth instead of residual error?