UCSB

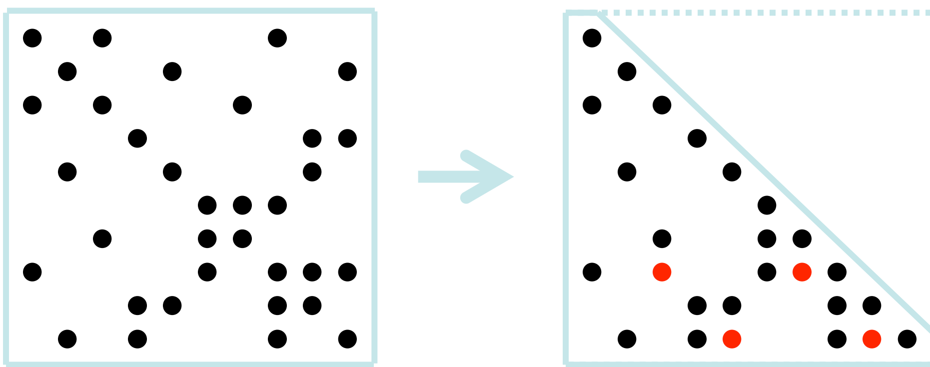
Empirical Complexity of Laplacian Linear Solvers: Discussion

Erik Boman, Sandia National Labs
Kevin Deweese, UC Santa Barbara
John R. Gilbert, UC Santa Barbara

Simons Institute Workshop on
Fast Algorithms via Spectral Methods
December 2, 2014

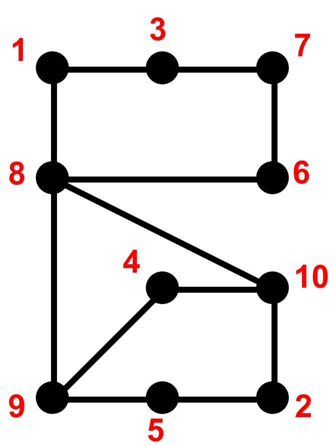
Support: Intel, Microsoft, DOE Office of Science, NSF

Graphs and sparse matrix computation (philosophical digression)

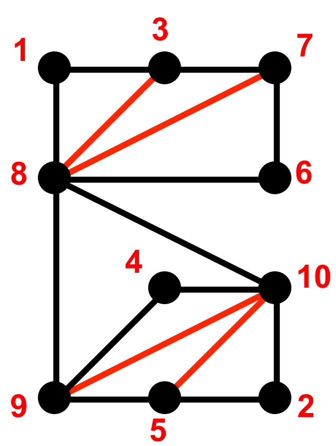


Symmetric Gaussian elimination:

for j = 1 to n
    add edges between j's
    higher-numbered neighbors



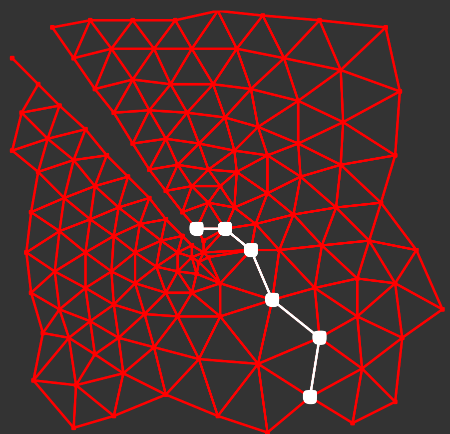
A



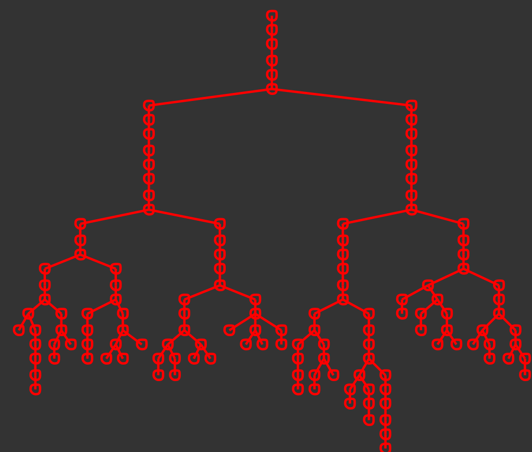
L (chordal)

Nested dissection and graph partitioning

[George 1971, then many papers]

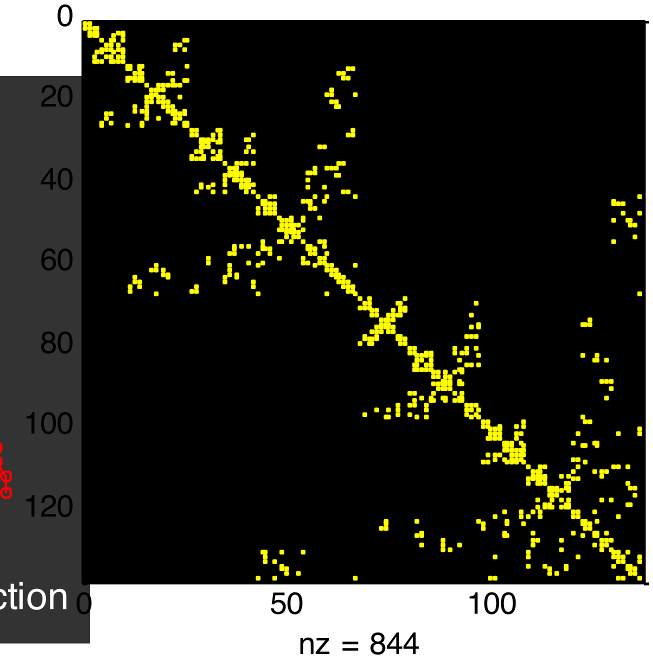


Vertex separator in graph of matrix



Elimination tree with nested dissection

Matrix reordered by nested dissection



- Find a small vertex separator, number it last, recurse on subgraphs
- Approx optimal separators => approx optimal fill & flop count

Graph algorithms in sparse matrix computation

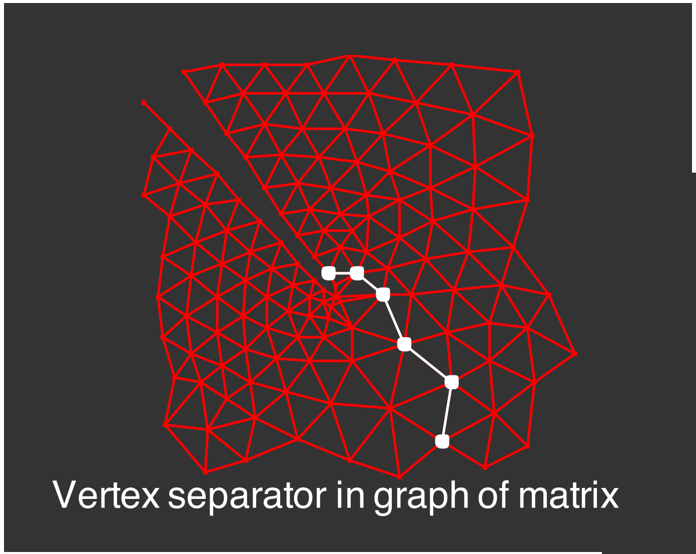
Many, many graph algorithms have been used, invented, and implemented at large scale for sparse matrix computation:

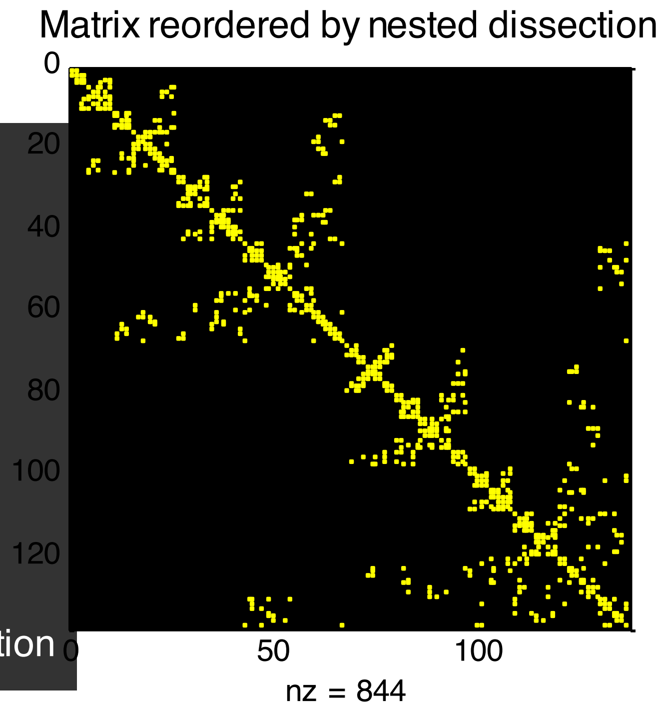
- Symmetric problems: elimination tree, nonzero structure prediction, sparse triangular solve, sparse matrix-matrix multiplication, min-height etree, …
- Nonsymmetric problems: sparse topological solve, bipartite matching (weighted and unweighted), Dulmage-Mendelsohn decomposition / strong components, …
- Iterative methods: graph partitioning again, independent sets, low-stretch spanning trees, …

UCSB

Nested dissection and graph partitioning

[George 1971, then many papers]





- Find a small vertex separator, number it last, recurse on subgraphs
- Approx optimal separators => approx optimal fill & flop count
- **It took more than 20 years for nested dissection to become the method of choice for sparse GE *in practice*.**

UCSB

Combinatorial Laplacian algorithms
(partial list)

- Vaidya 1990: $O(n^{1.75})$
- Spielman/Teng 2004: $O(n \log^c n)$
- Koutis/Miller/Peng 2010: $O(n \log n \log \log n)$
- Kelner/Orecchia/Sidford/Zhu 2013: $O(n \log^2 n \log \log n)$

(for sparse graphs, fixed ε)

UCSB

Combinatorial Laplacian algorithms
(partial list)

- Vaidya 1990: $O(n^{1.75})$
- Spielman/Teng 2004: $O(n \log^c n)$
- Koutis/Miller/Peng 2010: $O(n \log n \log \log n)$
- Kelner/Orecchia/Sidford/Zhu 2013: $O(n \log^2 n \log \log n)$

(for sparse graphs, fixed ε)

Okay, it's been more than 20 years now …

UCSB

The Laplacian World Championships

- Why should you participate?
- What would make you more likely to?
- How can we incent team leaders, members, HPC experts, … ?
- What are the right metrics? Single-core time; parallel time; anything not based on timings?
- Can “you implement, we measure” work for anything besides just single-core time?
- Thoughts on test graphs? Collections, generators, graphs supplied by contestants?
- Should code from entries be available for wider use?
- What’s the right schedule?
- What’s the right venue for the workshop?
- Other comments? Questions?
- How can we make this succeed?

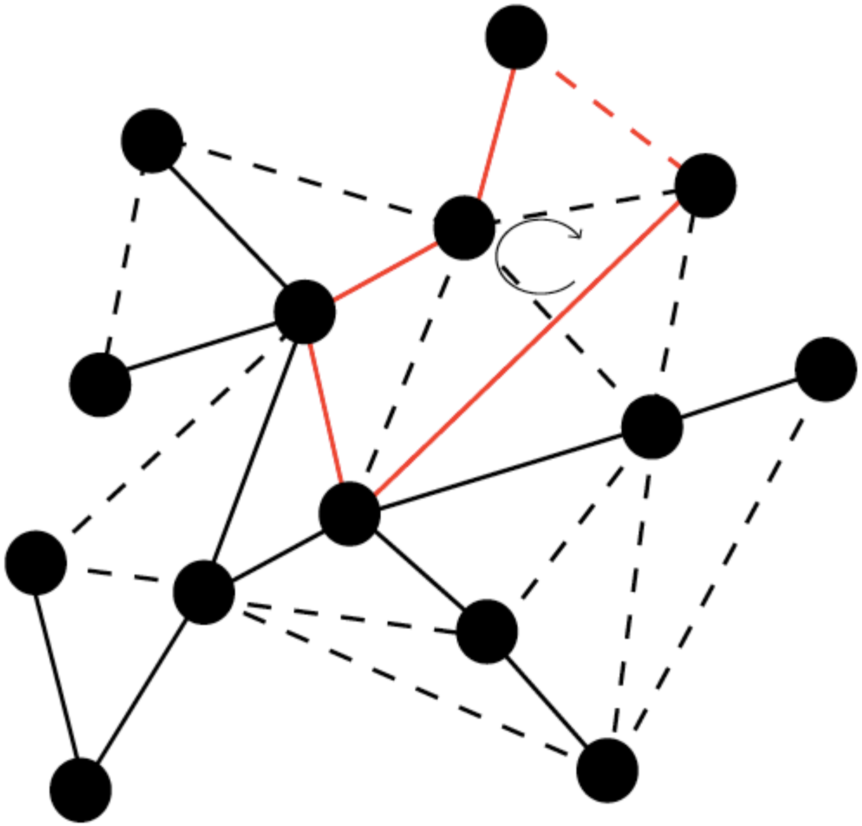
UCSB

Comparing empirical complexity to theory

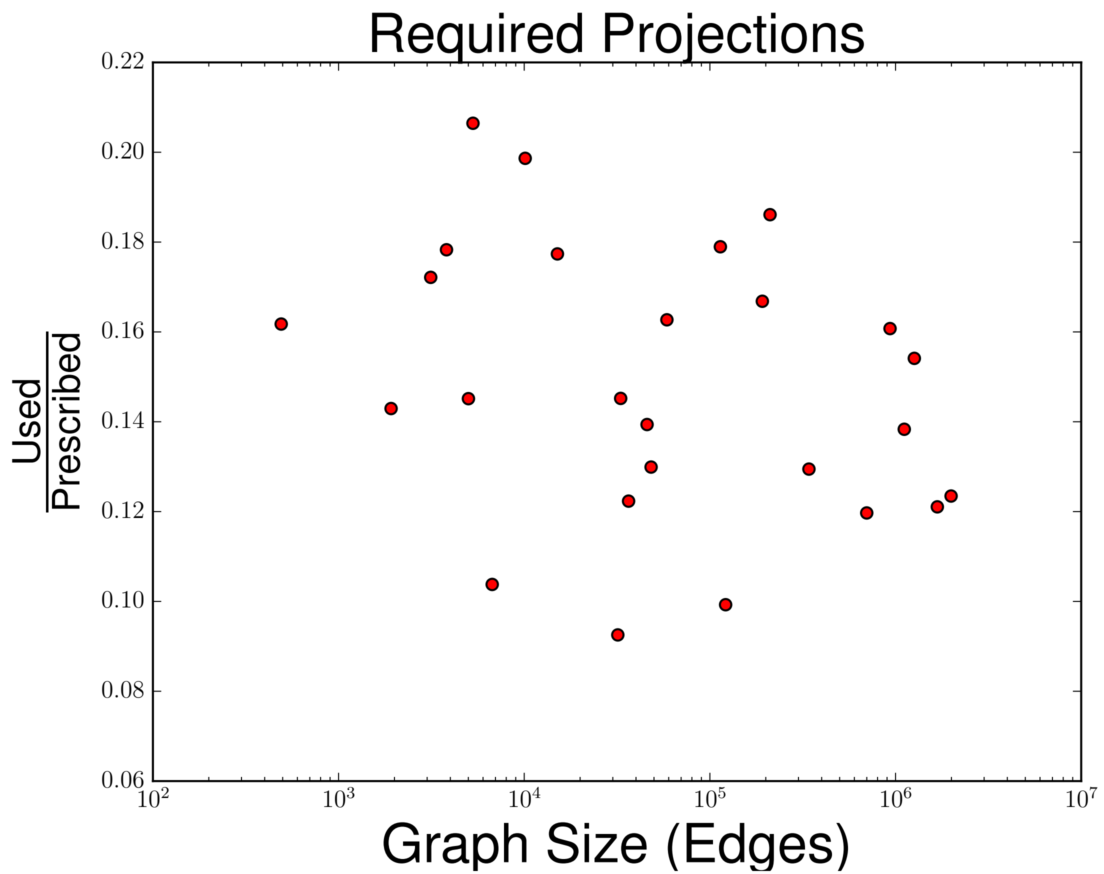
- Just counting things, not measuring time
  - a stepping stone between O( ) and running time
- Here: K/O/S/Z SimpleSolver (henceforth “RK”)
  - log n off of their FullSolver
- A whole bunch of sample matrices
  - UF collection, DIMACS, BTER, etc.

Kelner/Orecchia/Sidford/Zhu algorithm

- Select cycle (with probability proportional to stretch) from a fundamental cycle basis.
- Update flows around cycle.

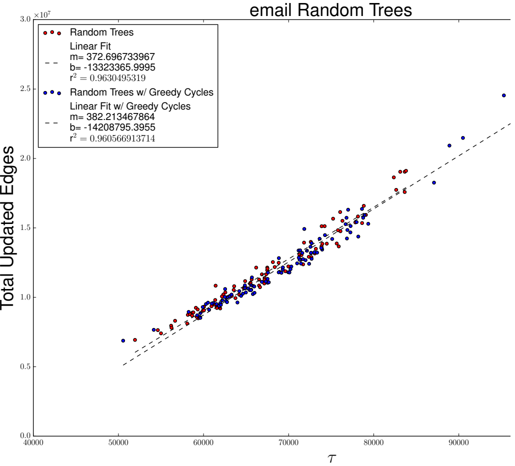


Number of projections to convergence

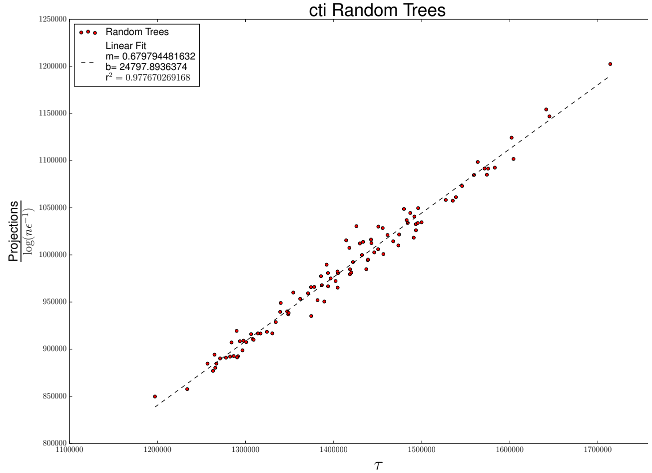


- More optimistic than the bound by a factor of 5 or 10.

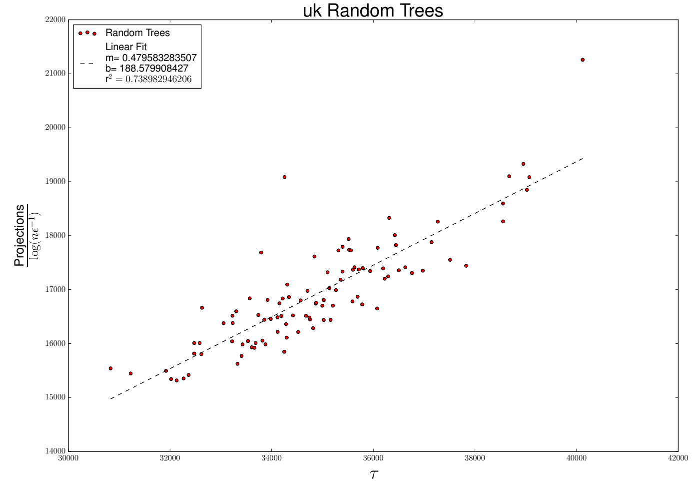
Scaling of work with tree stretch



email interactions



physical model



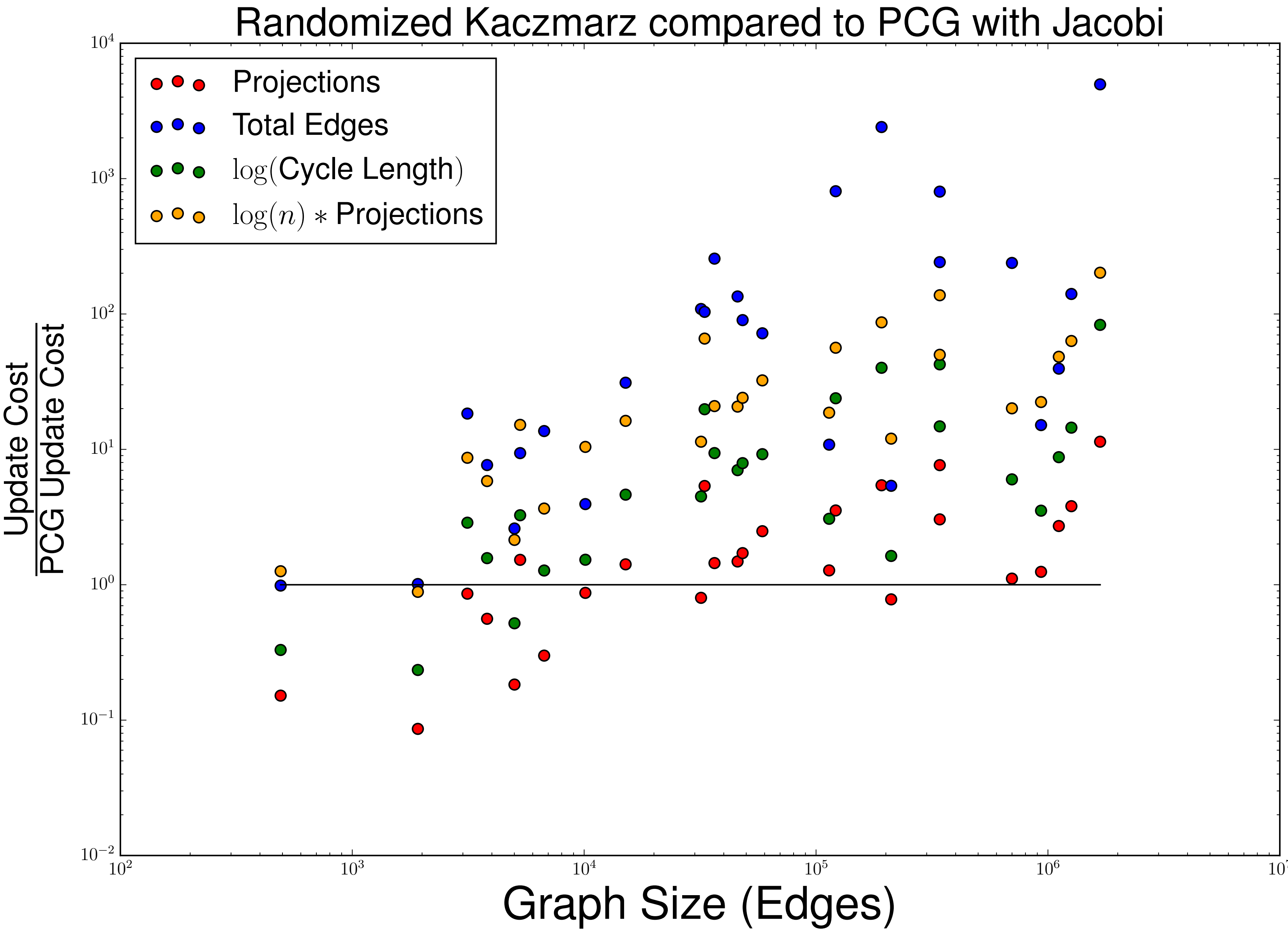
road network

- Good correlation, better for some types of graphs than others.

Empirical complexity of RK compared to PCG

- PCG = conjugate gradient preconditioned by Jacobi
- PCG does about iters * (m+n) "edge touches"
- Count the cost of an RK projection in four different ways:
  - cycle length (naive)
  - lg n (fast fundamental cycle data structure)
  - lg (cycle length) (unwarranted optimism)
  - 1 (surely a lower bound)

UCSB



Randomized Kaczmarz compared to PCG with Jacobi
Projections
Total Edges
log(Cycle Length)
log(n) * Projections
$\frac{\text{Update Cost}}{\text{PCG Update Cost}}$
Graph Size (Edges)
10^{4}
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}
10^{7}

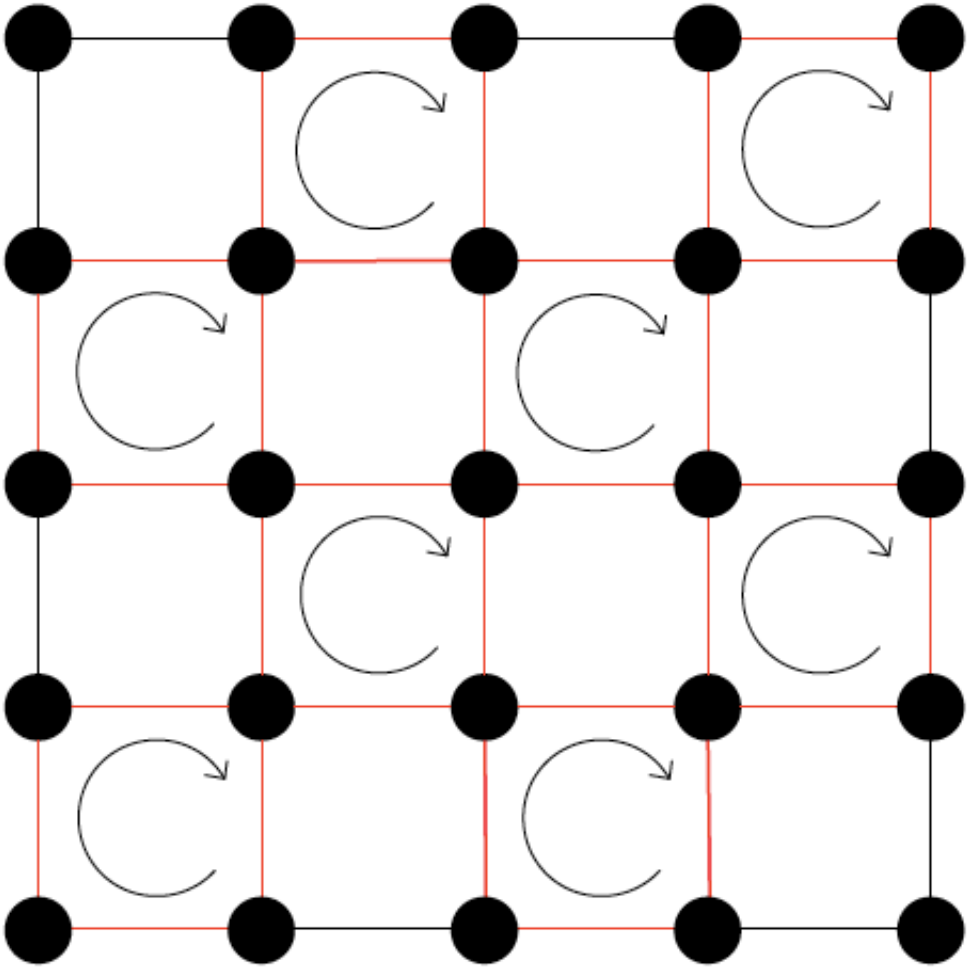
How about expanding the available projections?

- Add some cycles to basis
- Potentially gain parallelism in updates
- Maybe add flexibility with non-fundamental cycles

- More of the projection cost metrics are fantasy now, since we don't have fast update algorithms in general.

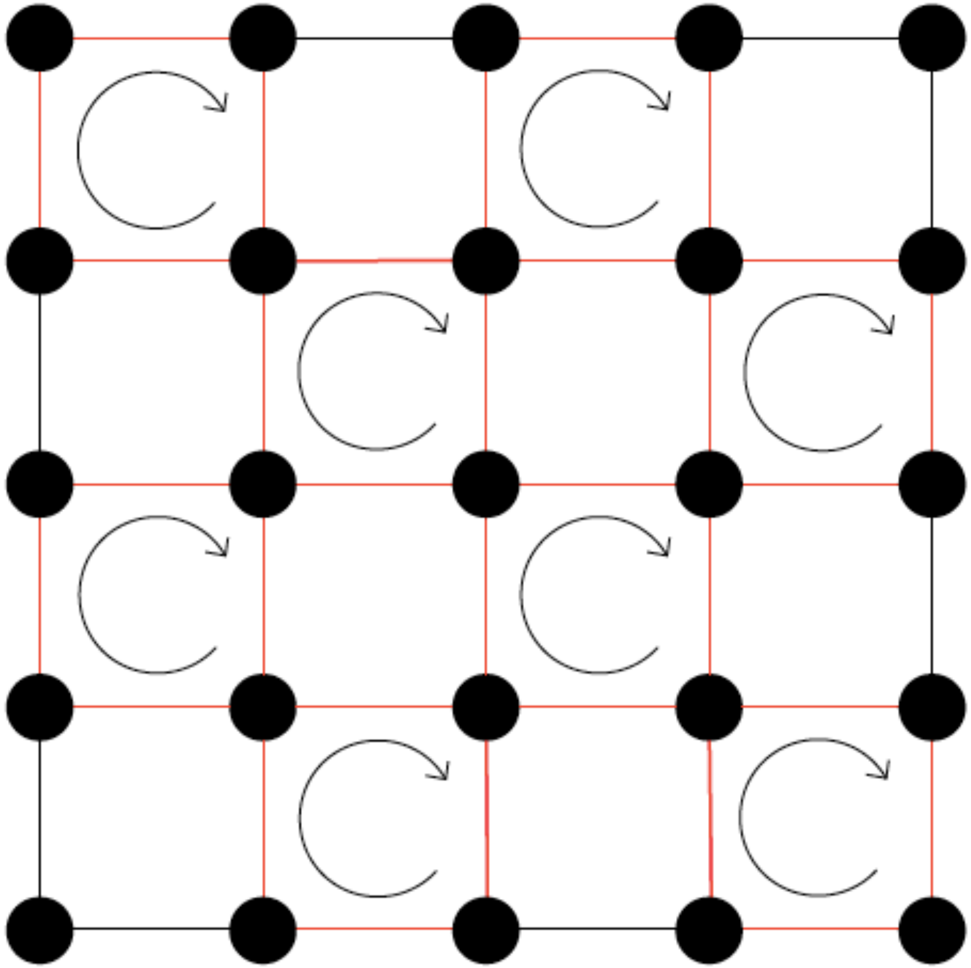
Example: non-fundamental cycles on a square grid

- Edge-disjoint cycles can be updated in parallel.

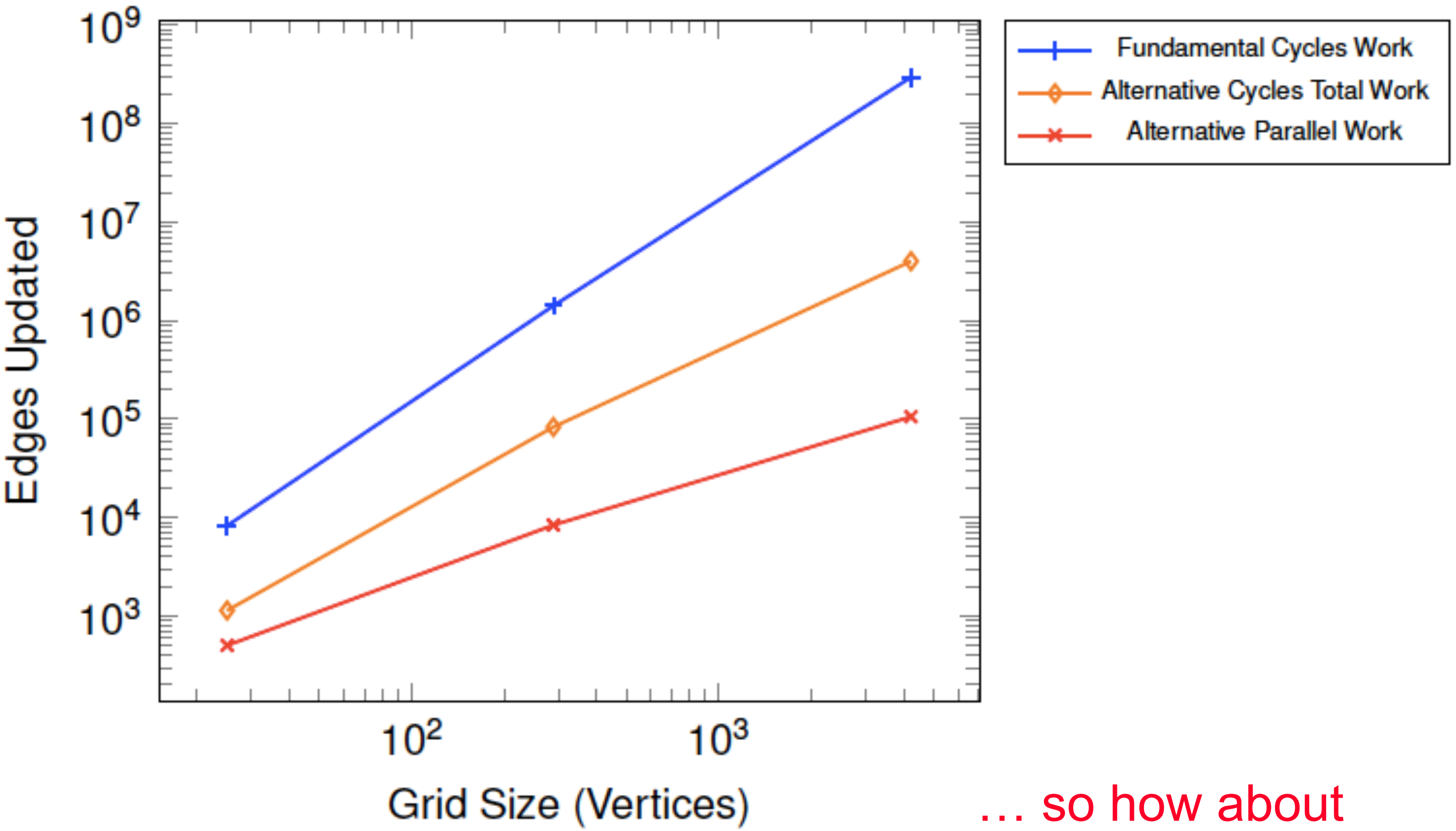


Example: non-fundamental cycles on a square grid

- Edge-disjoint cycles can be updated in parallel.

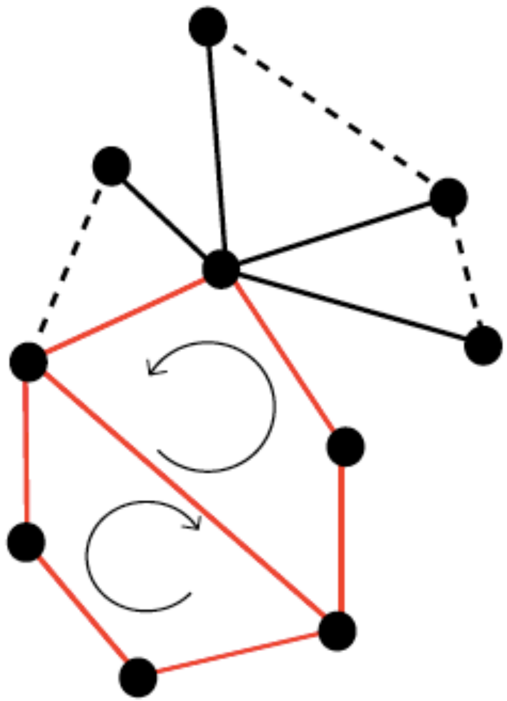


Work and span for square grid example



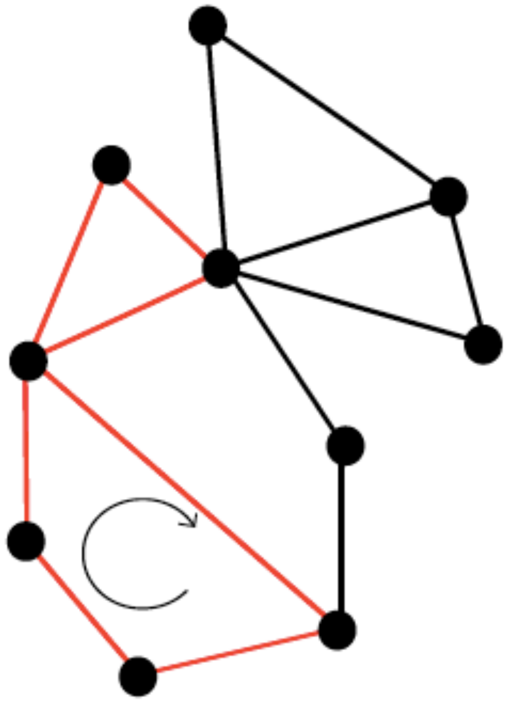
… so how about general graphs?

Tree shortcut cycles



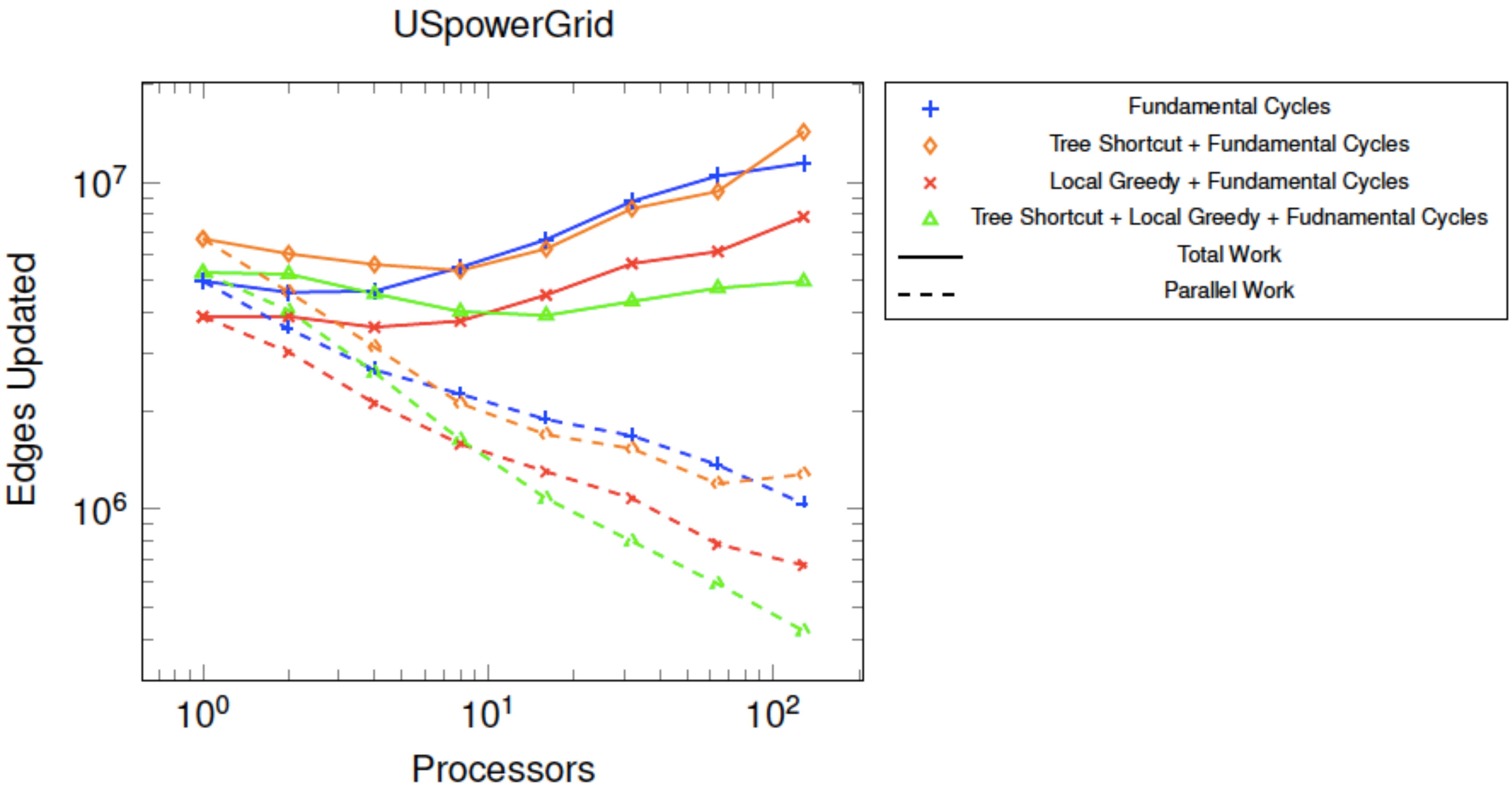
- Select an off-tree edge.
- Search for a shortcut between endpoints of this edge.
  - Only search edges closer to root of tree.
  - Truncate search to control cost.
- Replace tree cycle with shortcut cycle.
  - Cycle space dimension remains the same.

Local greedy cycles

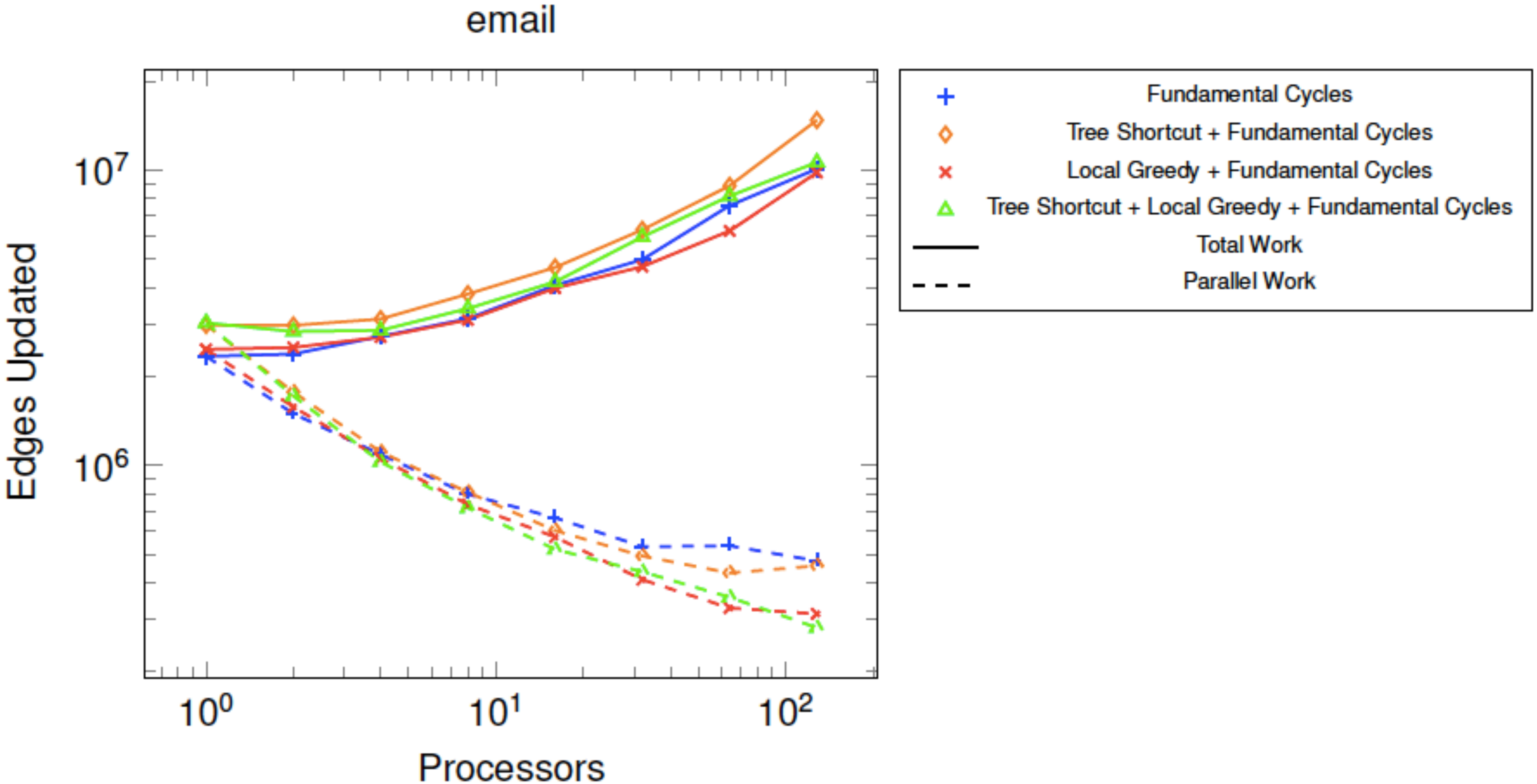


- Select an unmarked edge.
- Find smallest cycle containing this edge.
  - Truncate search to control cost.
  - If found mark all the used edges.
- This set is not guaranteed to span the cycle space.

Comparing cycle sets: work and parallel work



Comparing cycle sets: work and parallel work



The Laplacian World Championships

- Why should you participate?
- What would make you more likely to?
- How can we incent team leaders, members, HPC experts, … ?
- What are the right metrics? Single-core time; parallel time; anything not based on timings?
- Can “you implement, we measure” work for anything besides just single-core time?
- Thoughts on test graphs? Collections, generators, graphs supplied by contestants?
- Should code from entries be available for wider use?
- What’s the right schedule?
- What’s the right venue for the workshop?
- Other comments? Questions?
- How can we make this succeed?