A Parallel Solver for Laplacian Matrices

Tristan Konolige (me) and Jed Brown

University of Colorado
Boulder

Graph Laplacian Matrices

- Covered by other speakers (hopefully)
- Useful in a variety of areas
- Graphs are getting very big
  - Facebook now has ~couple billion users
  - Computer networks for cyber security
- Interested in network graphs
  - Undirected
  - Weighted
- We will need faster ways to solve these systems
- Note: Laplacians have constant vector as nullspace

Why Parallelism

- Graphs are growing but single processor speed is not
- Want to process existing graphs faster or do larger network analysis
- Clock speed has stagnated
  - Bandwidth increasing slowly
- Processor count/machine count growing
  - Xeon Phi, etc.
- Going to look at distributed memory systems
  - Most supercomputers and commodity clusters

Goals

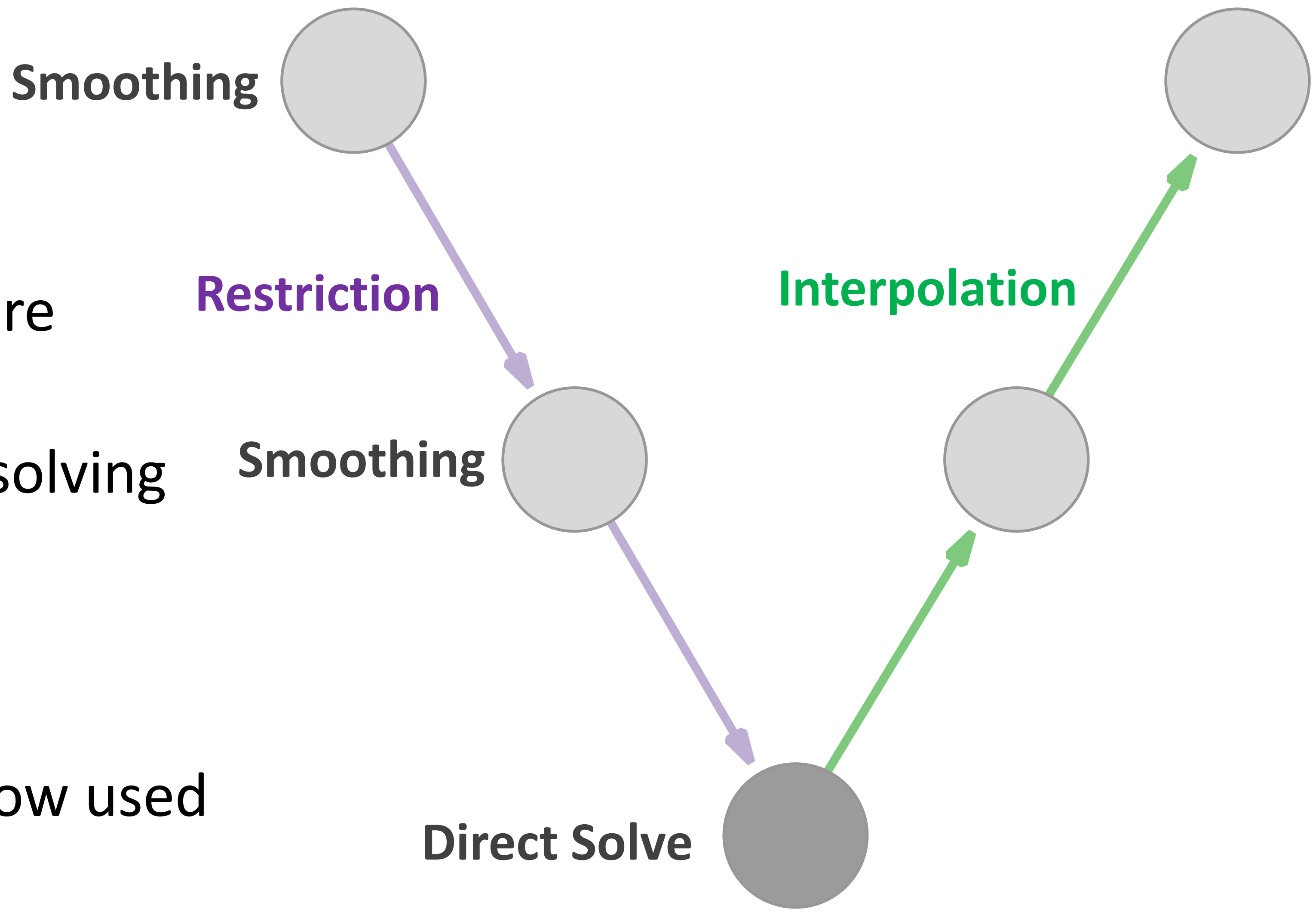
- Parallel scalability out to large numbers of processors/nodes
- Convergence factors close to LAMG
- Interested mostly in scale-free graphs for now

Existing Solvers

- Spielman and Teng’s theoretical nearly-linear time solver
  - No viable practical implementations
  - Many other theoretical solvers
- Kelner solver (previous talk w/ Kevin)
- Combinatorial Multigrid from [Koutis and Miller]
- Lean Algebraic Multigrid from [Livne and Brandt]
- Degree Aware Aggregation from [Napov and Notay]
- CG a variety of preconditioners
- Direct solvers

Multigrid

- Both CMG and LAMG are multigrid solvers
- Multilevel method for solving linear systems
- O(N) (ideally)
- Originally intended for geometric problems, now used on arbitrary matrices



A V-cycle

Lean Algebraic Multigrid [Livne and Brandt 2011]

- Low degree elimination
  - Eliminate up to degree 4
  - Reduces cycle complexity
  - Incredibly useful on network graphs
- Aggregation based Multigrid
  - Restriction/interpolation from fine grid aggregates
  - Avoids aggregating high-degree nodes
  - Based on strength of connection + energy ratio
  - Typically smoothed restriction/interpolation

LAMG

- Caliber 1 interpolation (unsmoothed restriction/interpolation)
  - Avoids complexity from fill in
- Gauss-Seidel Smoothing
- Multilevel iterant recombination – adaptive energy correction
  - Similar to Krylov method at every level
- O(N) empirically

LAMG

- Hierarchy alternates between elimination and aggregation
- First level elimination only applied once during solve

Level	Size	NNZ	Type	Time (s)	Comm Size	Imb
0	1069126	113682432	Elim	0.1180	64	1.10
1	1019470	113385358	Reg	0.7480	64	1.11
2	75493	18442801	Elim	0.0090	64	1.46
3	62072	18374722	Reg	0.0687	64	1.23
4	8447	1265927	Elim	0.0016	64	2.87
5	5153	1250659	Reg	0.0052	64	1.49
6	466	20188	Elim	0.0004	1	1.00
7	173	19125	Reg	0.0019	1	1.00
8	18	56	Elim	0.0001	1	1.00
9	3	7	Reg	0.0001	1	1.00

Implementation

- C++ and MPI
  - No OpenMP for now
- CombBLAS for 2D matrix decomposition [Buluç and Gilbert 2011]
  - Needed for scaling
  - Helps distribute high-degree hubs
- Randomized matrix ordering
  - Worse locality
  - Greatly improves load balance
- Jacobi Smoothing
- V-cycles
  - No iterant recombination, requires multiple dot-products which are slow in parallel
  - Instead use constant correction
- CG preconditioner
  - Worse than energy correction
  - Orthangonalize every cycle
- Manually redistribute work if problem gets too small

Parallel Low-Degree Elimination

- Difficult part is if there are two low-degree neighbors
- Can’t eliminate both at once
- Use SpMV to choose which neighbors to eliminate
    - Boolean vector indicating degree < 4
    - Semiring is {min(hash(x), hash(y)), id}
- Can use multiple iterations to eliminate all low-degree nodes
    - In practice, one iteration eliminates most low-degree nodes

Parallel Aggregation

```
for each undecided node n:
  let s = undecided or seed neighbor with
          strongest connection and not full
  if s is a seed:
    aggregate n with s
  if s is undecided:
    s becomes a seed
    aggregate n with s
end
```

- Aggregates depend on order

Parallel Aggregation

- SpMV iterations on strength of connection matrix to form aggregates
  - Vector is status of node {Undecided, Aggregated, Seed,FullSeed}
  - Semiring + is max (i.e. strongest connection)
  - x * y is y if x == Undecided or Seed otherwise 0
  - In resulting vector, if x found an Aggregated vertex, we aggregate. Otherwise x votes for is best connection
  - Undecided nodes with enough votes are converted to seeds
  - <10 iteration before every node is decided
- Cluster size is somewhat constrained
  - As long as clusters have a reasonable size bound, results are fine
- We do not use energy ratios in aggregation (yet)
  - Will have worse aggregates than LAMG

Strength of Connection

- LAMG uses a strength of connection metric for aggregation
  - Relax on Ax=0 for random x
- In our tests, algebraic distance [Safro, Sanders, Schulz 2012] performs slightly better than affinity
  - 58.49% of fastest solves used algebraic distance vs 41.51% with affinity

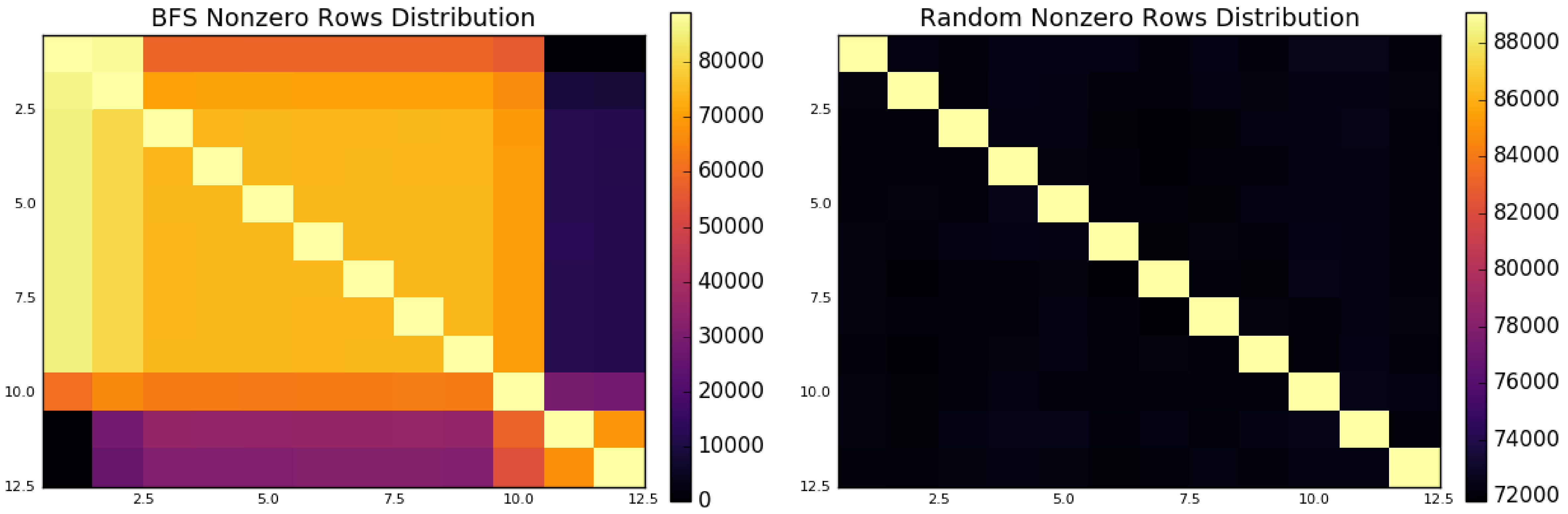
$$c_{uv} := \frac{\left|(X_u, X_v)\right|^2}{(X_u, X_u)^2 (X_v, X_v)^2} \quad (X, Y) := \sum_{k=1}^{K} X^{(k)} Y^{(k)}$$

Affinity

$$\rho_{ij} = \Big(\sum_{r=1}^{R} |\chi_i^{(k,r)} - \chi_j^{(k,r)}|^2 \Big)^{\frac{1}{2}}$$

Algebraic distance

Matrix Randomization



Results

- All tests run on NERSC's Edison
  - 2x 2.4GHz 12-core Intel "Ivy Bridge" processor per node
  - Cray Aries interconnect
  - 4 MPI tasks per node
- LAMG Serial implementation by [Livne and Brandt]
  - In MATLAB with C mex extensions
- Solve to 1e-8 relative residual norm
- Code is not well optimized
- Interested in scaling

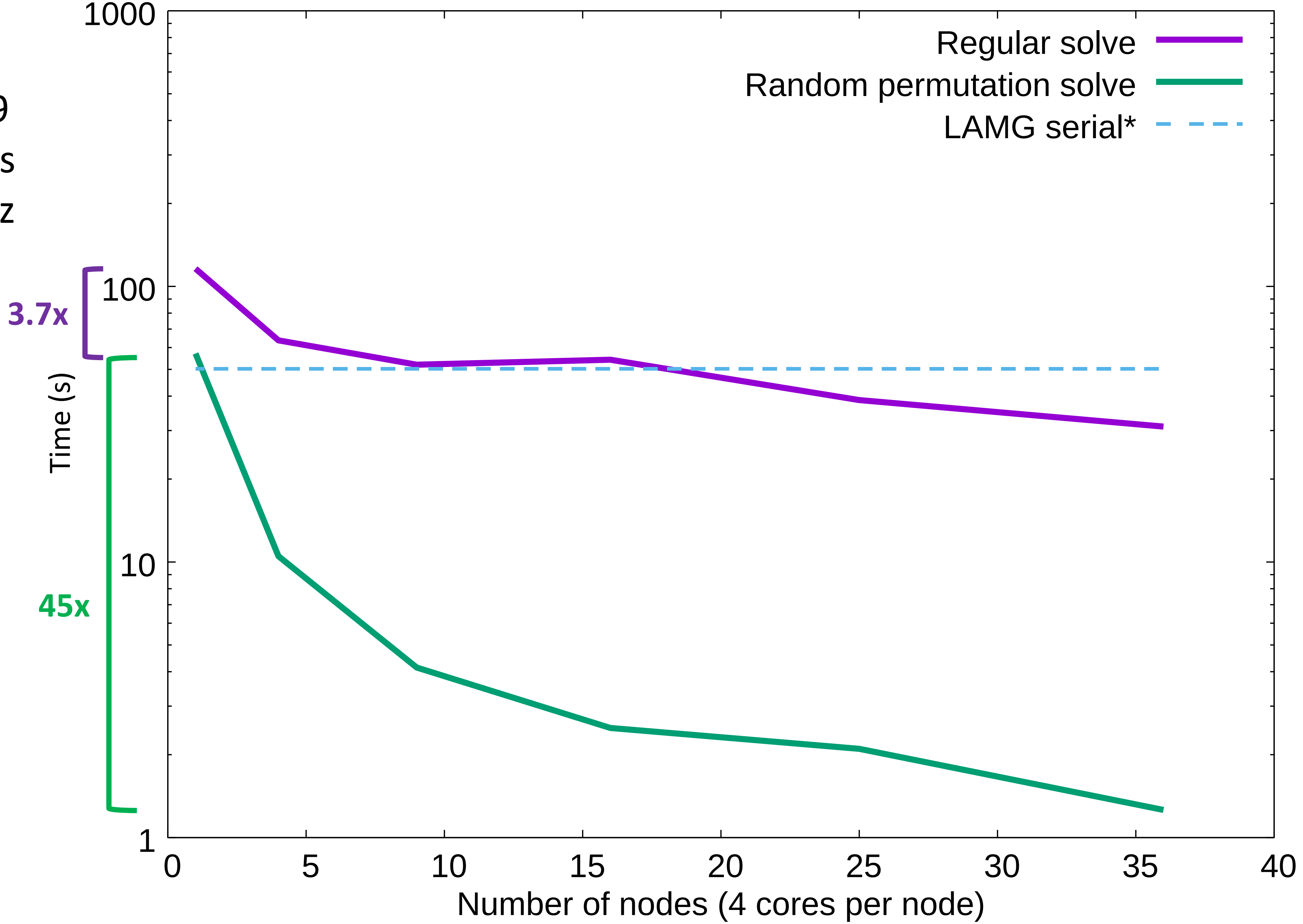
Convergence Factors

- Cycle complexity: nnz(all ops)/nnz(finest matrix)
- Effective Convergence Factor (ECF) Δ ||residual || ^ 1/cycle complexity

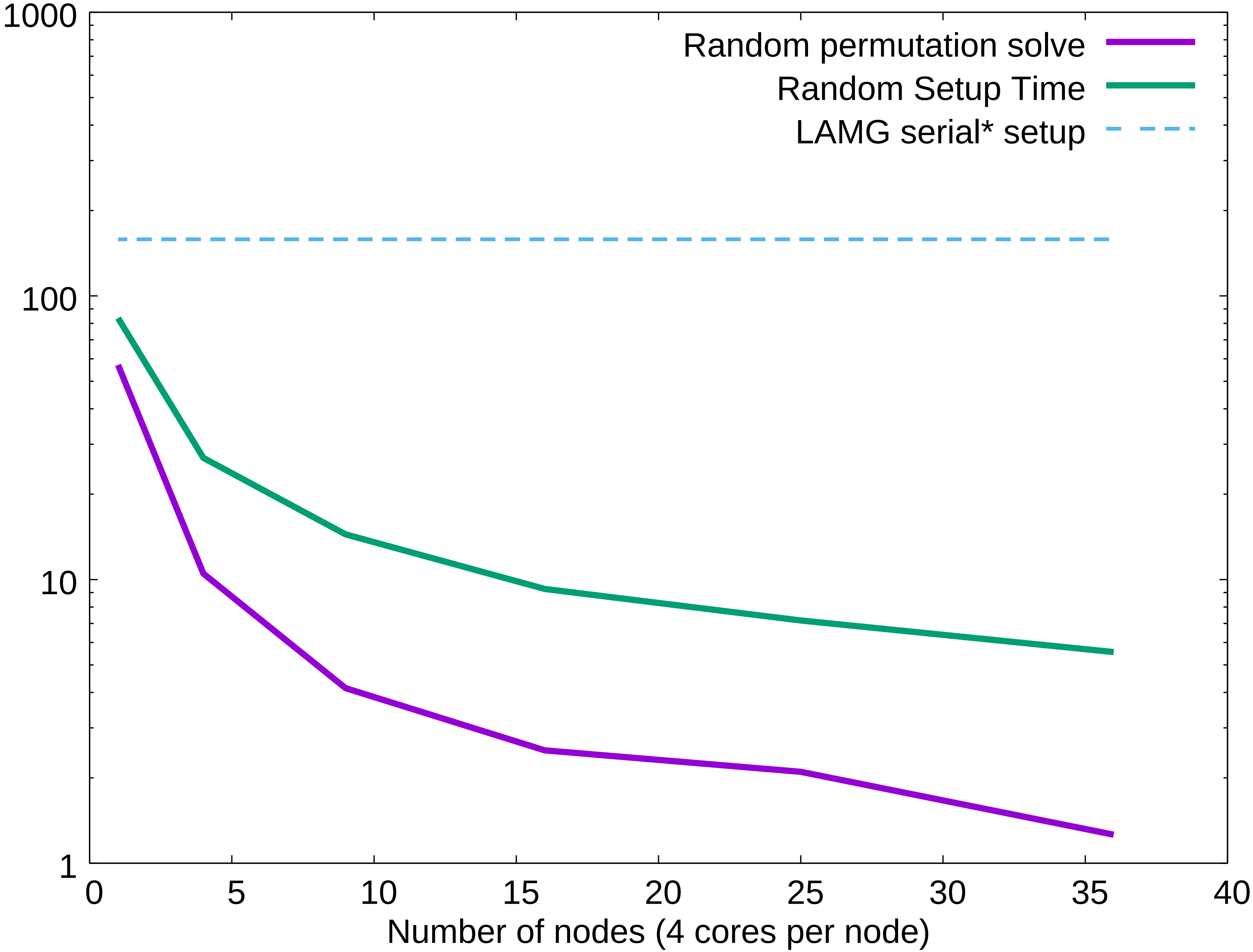
Matrix	ECF Serial LAMG	ECF Our Solver	ECF Jacobi PCG
hollywood-2009	0.540	0.856	0.992
citationCiteseer	0.816	0.919	0.938
astro-ph	0.695	0.800	0.846
as-22july06	0.282	0.501	0.784
delaunay_n16	0.812	0.896	0.980

- No GS-smoothing
- No iterant recombination
- Poorer aggregates

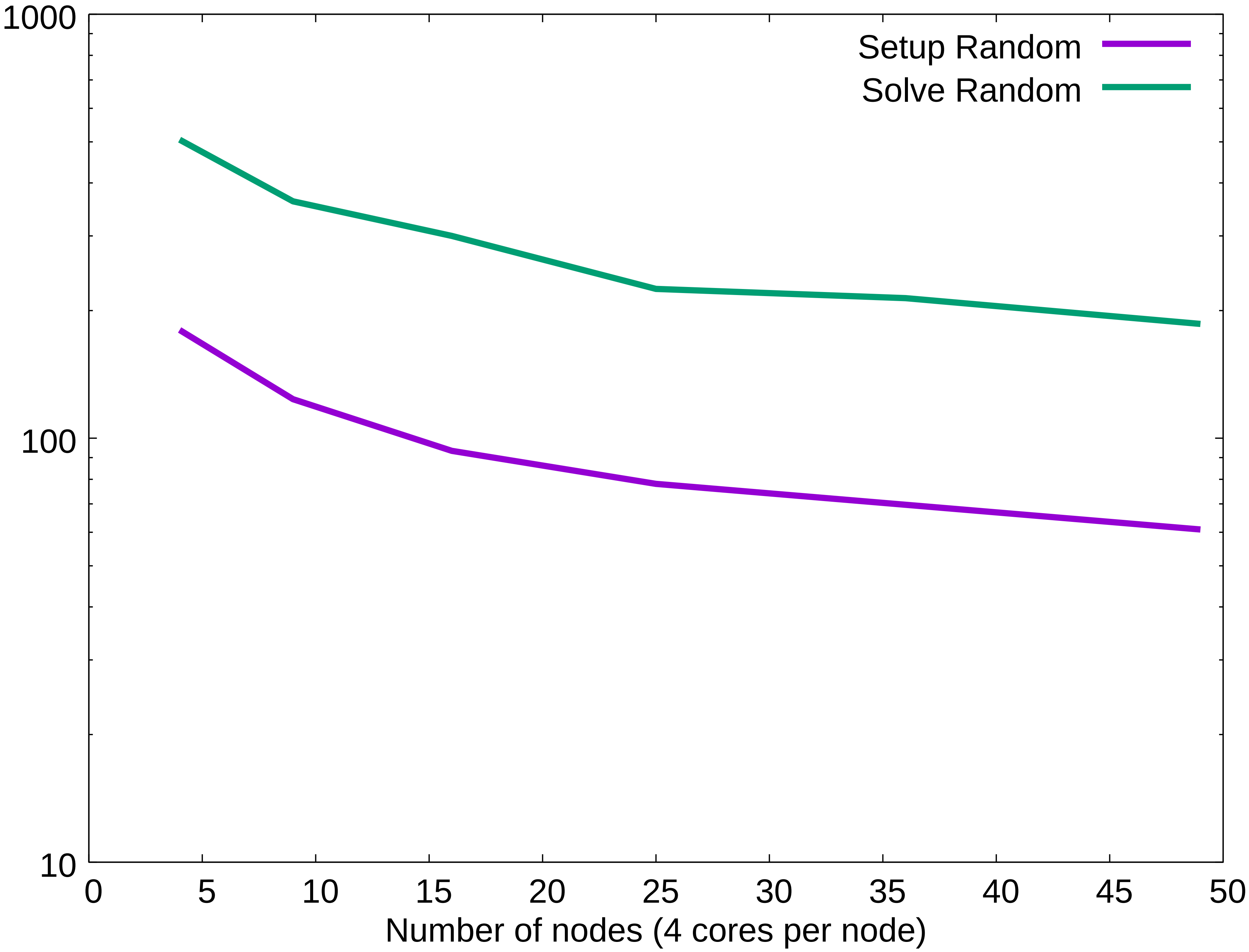
hollywood-2009
1,139,905 nodes
113,891,327 nnz



hollywood-2009
1,139,905 nodes
113,891,327 nnz



europe_osm
rows 50,912,018
nnz 108,109,320



Conclusion & Future Work

- Distributed memory solver show significant speedups
  - Even without complex aggregation strategies
- Matrix randomization provides large benefit
- Improve aggregation with energy ratios
  - Convergence rates still well below LAMG
  - Particular graphs have very poor rates

Thank you