Exceptional service in the national interest

Sandia National Laboratories

Solving Graph Laplacians for Complex Networks

SIAM LA15, Atlanta, Oct. 2015

Erik Boman, Sandia National Labs

Kevin Deweese and John R. Gilbert, UCSB

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Outline

- Graph Laplacians
- Linear systems and preconditioners
- Normalization
- Empirical study with Trilinos
- Nearly-optimal combinatorial solvers
  - Kelner et al.'s simple iterative method
- Conclusions

Complex Networks: Numerical Computing

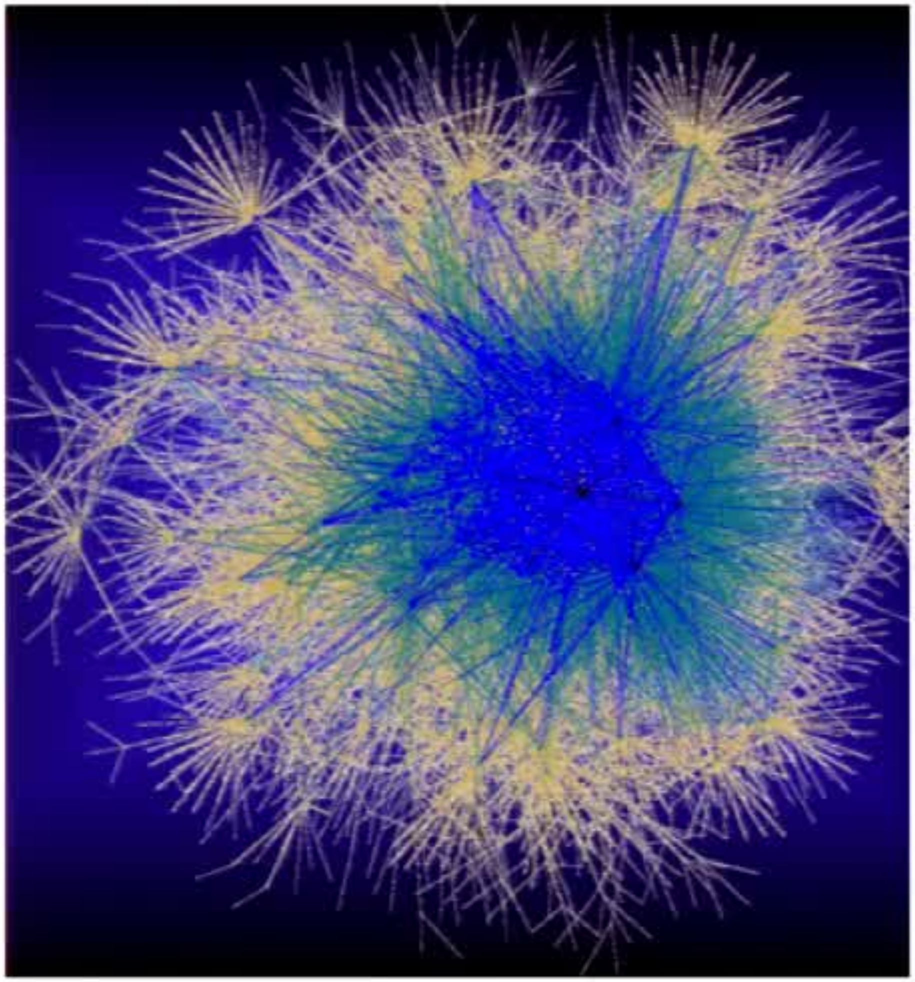
Complex networks often analyzed by

- Degree distribution
- Clustering coefficient
- Centrality metrics

Less attention on numerical linear algebra:

- Linear system: $Ax=b$
- Eigenproblem: $Ax = \lambda x$

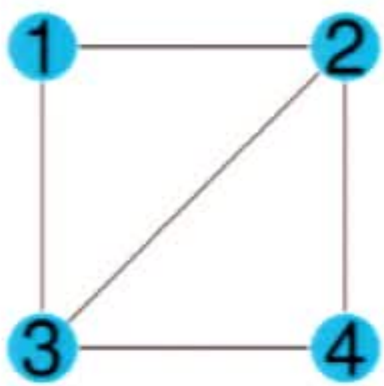
Well studied for PDEs, but not for complex networks.



BGP graph (credit: Richardson, Chung)
http://math.ucsd.edu/~fan/graphs/gallery

Matrices from Graphs

Symbol	Matrix
A	Adjacency matrix
D	Diagonal vertex degree matrix
L = D-A	Graph Laplacian
$L_N = D^{-1/2}(D-A)D^{-1/2}$	Normalized Laplacian



$$\begin{pmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{pmatrix}$$

Solving Linear Systems

Different research communities, different approaches!

- Numerical linear algebra
  - Empirical focus
    - Analysis for model problems sufficient
  - Main application: discretizations of PDEs
  - Good and robust software for solving large systems
- CS Theory
  - Focus on theory and complexity
    - Worst-case analysis
  - Main target: graph Laplacians, SDD systems
  - Software not important (some Matlab codes)
- Network Science
  - Just a tool – don't care how it's done

Solvers and Preconditioners

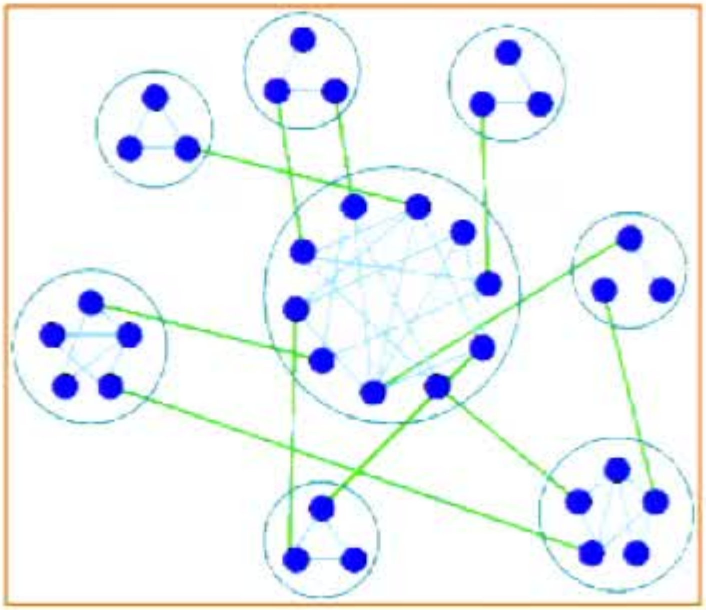
- Sparse direct factorization only viable for small problems
- Stationary iterations (Jacobi, Gauss-Seidel) converge but quite slowly
- Conjugate gradients or Chebyshev acceleration reduces #iterations.
  - Key is to find good preconditioner M≈A
- Classic "black-box" algebraic preconditioners:
  - Jacobi (diagonal)
  - Symmetric Gauss-Seidel (SGS)
  - Incomplete Cholesky (IC)
- Algebraic multigrid (AMG)
  - Developed for PDEs on meshes, not complex networks
  - Recent progress tuning for complex networks (LAMG)

BTER Graph Generator

How to generate realistic graphs/networks?

We use BTER: Block Two-level Erdös-Renyi

- Kolda, Pinar, Seshadri (2014)
- Captures skewed degree distributions
    - Not necessarily power-law
- Has community structure
- Able to "fit" real data
    - Degree distribution
    - Clustering coefficient



Experiments

- Study two groups of graphs
  - Real networks from UF and SNAP collections
    - Social networks, web graphs, collaboration networks, etc.
    - 25 graphs, up to 735K vertices (3.5M edges)
  - Synthetic graphs (BTER)
    - Log normal degree distribution, but vary sizes and avg. degree
- Solve singular Lx=b where the solution is a random vector, using projected PCG
  - Null-space is just the constant vector
  - Use Trilinos software (next slide)
- Solvers have two phases
  - Setup (preconditioner setup or symbolic+numeric factorization)
  - Solve (CG iteration or triangular solves)

Trilinos Computational Science Toolkit

- Collection of ~60 packages
  - Heroux et al., Sandia
- Trilinos Capabilities:
  - Scalable Linear & Eigen Solvers
  - Discretizations, Meshes & Load Balancing
  - Nonlinear & Optimization Solvers
  - Software Engineering Technologies & Integration
- Parallel:
  - MPI for distributed memory
  - Growing support for shared-memory (OpenMP, pthreads, CUDA)

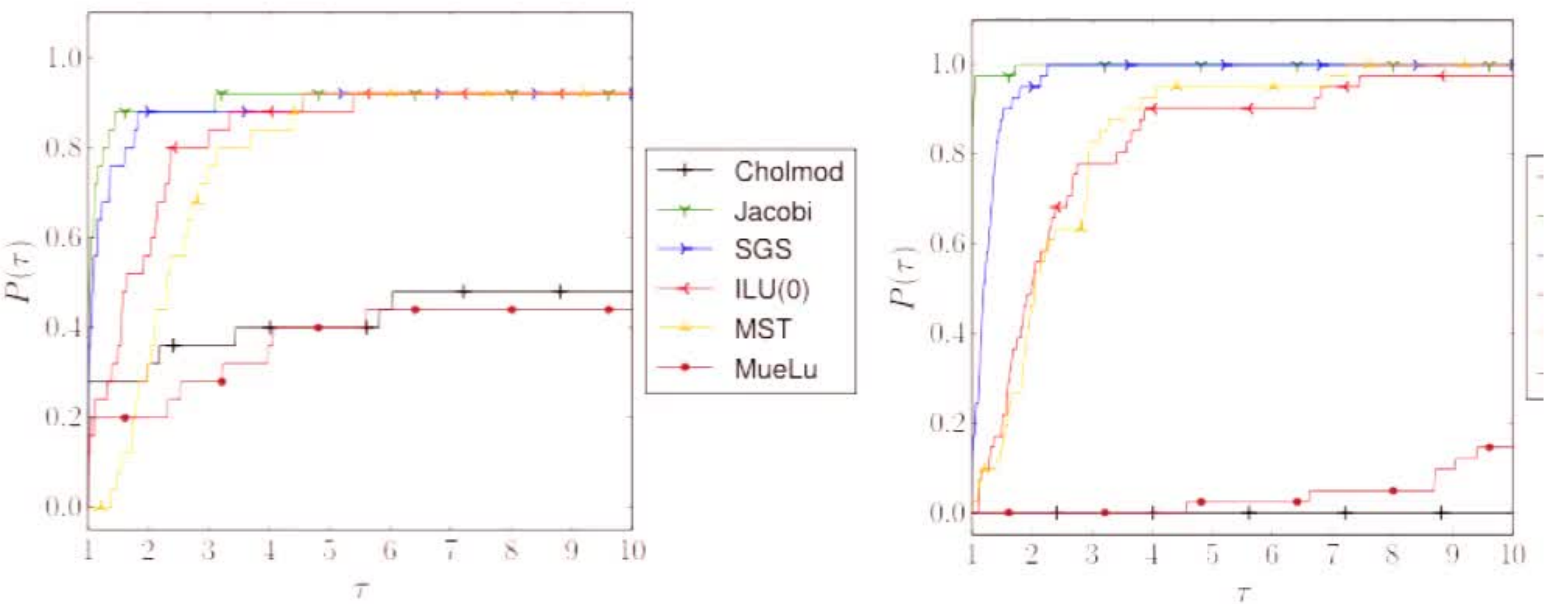
Packages we used:

- Tpetra: Matrices & vectors
- Belos: Iterative solvers
- Ifpack2: Preconditioners
  - Jacobi, Gauss-Seidel
  - Incomplete factorizations
  - Subgraph preconditioners
- MueLu: Multigrid

Performance Profile

Total time = Setup time + solve (iteration) time

- UF real networks
- BTER



Why do BTER differ from real graphs?

- BTER designed to match
  - Degree distribution
  - Clustering coefficient
  - Not eigenvalues!

- We tested BTER replica of Amazon-2008 network
  - Check #iterations for Laplacian solve

Graph	Cond.no.	Jacobi	SGS	ILU(0)	AMG
Original	1.5e5	3233	1290	1211	216
BTER	2.0e4	726	349	336	150

Combinatorial Preconditioners: Great in Theory

Core Idea: Construct a sparser graph that is a good spectral approximation (spectral sparsifier), use this as preconditioner.

- Typically, use a carefully chosen subgraph
  - For example, spanning tree + "a bit more"
- First proposed by Vaidya ('90, unpublished)
  - Described and analyzed in [Bern et al. '06], implemented by [Chen and Toledo, '03]
- Support theory extensions [B., Hendrickson, '03]
- A decade of improving complexity for Laplacian/SDD solvers
  - Significant work on "near optimal solvers"
    - Spielman & Teng ('04,'05), Koutis-Miller-Peng ('10,'11), others…
  - Kelner et al. ('13): dual randomized Kaczmarz
    - Lee & Sidford ('13): coordinate descent

Are They Competitive?

- Most combinatorial near-optimal solver/preconditioners are very complicated and have never been implemented
- The recent KOSZ/DRK method is simpler:
  - Solves a dual problem on the edges of the graph
  - Corresponds to flows in an electrical network
  - Randomly sample a cycle, update flow along edges, repeat
  - This is randomized Kaczmarz (on a dual problem)
  - No CG required as convergence is provably good without
- Two recent papers evaluate this method:
  - Hoske, Lukarski, Meyerhenke, Wegner (2015)
  - B., Deweese, Gilbert (2015)
  - Both conclude KOSZ/DRK is not competitive on unweighted graphs