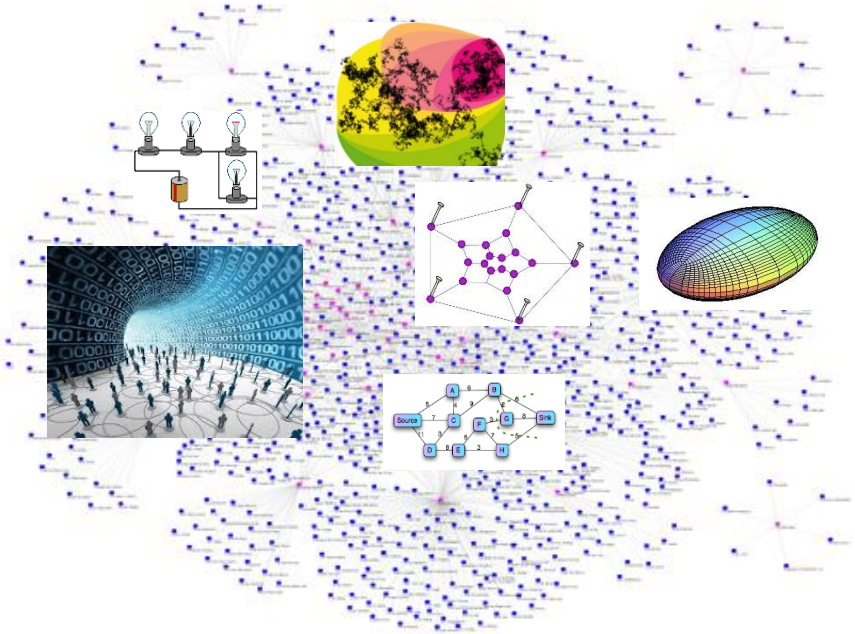
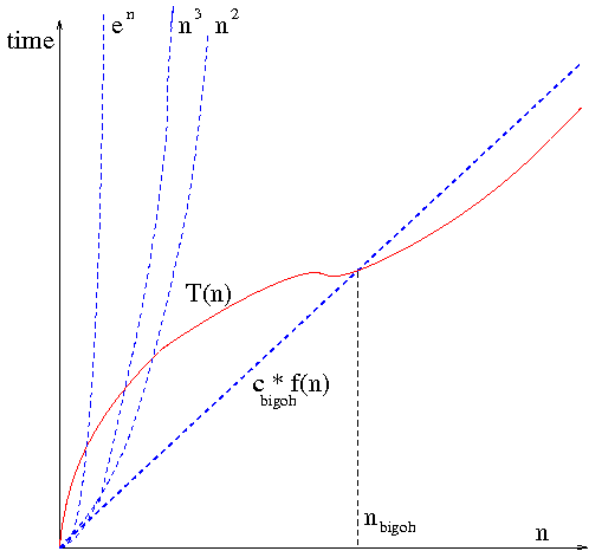
Scalable Algorithms in the Age of Big Data and Network Sciences

Shang-Hua Teng

USC



Asymptotic Complexity



$O(f(n))$

$\Omega(g(n))$

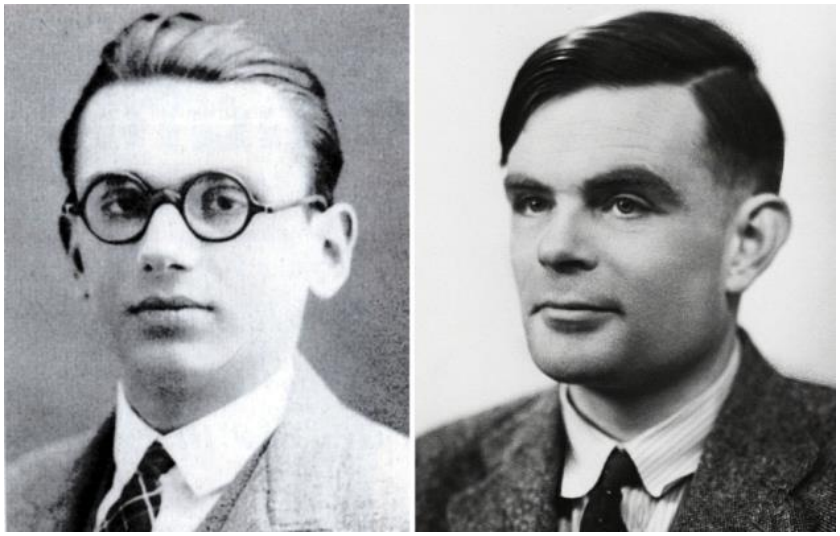
$\Theta(h(n))$

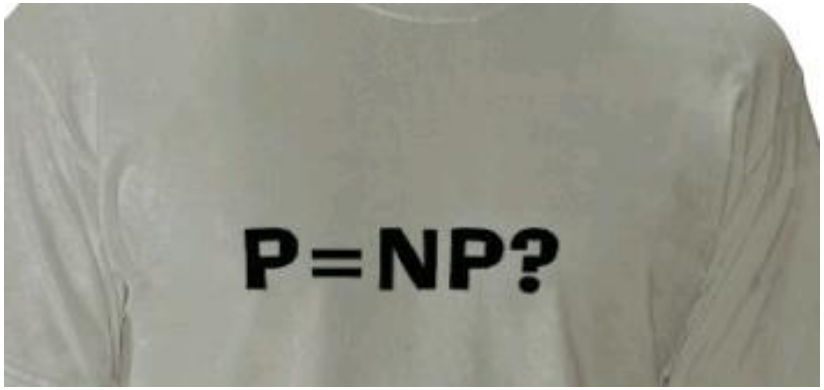
for problems with massive input

Characterization of Efficient Algorithms

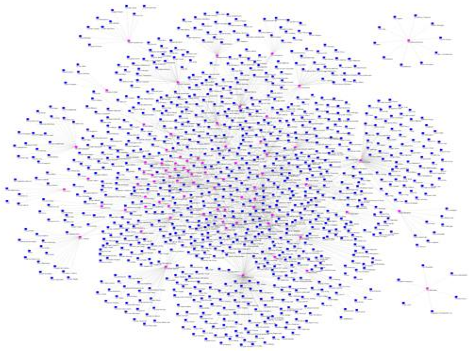
Polynomial Time

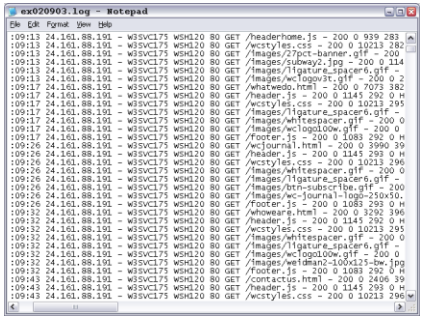
$O(n^c)$ for a constant c





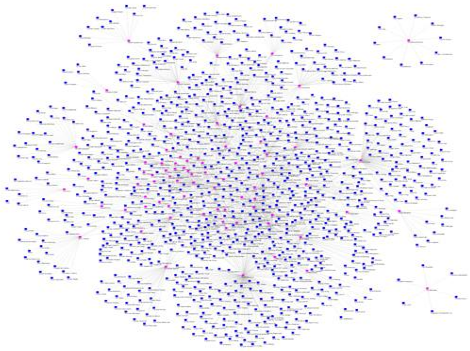
Big Data and Massive Graphs

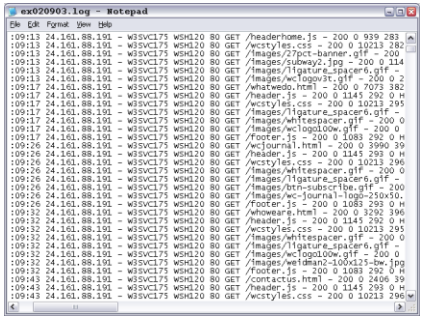




- Tera Web pages
- unbounded amount of Web logs
- billions of variables
- billions of transistors.

Big Data and Massive Graphs

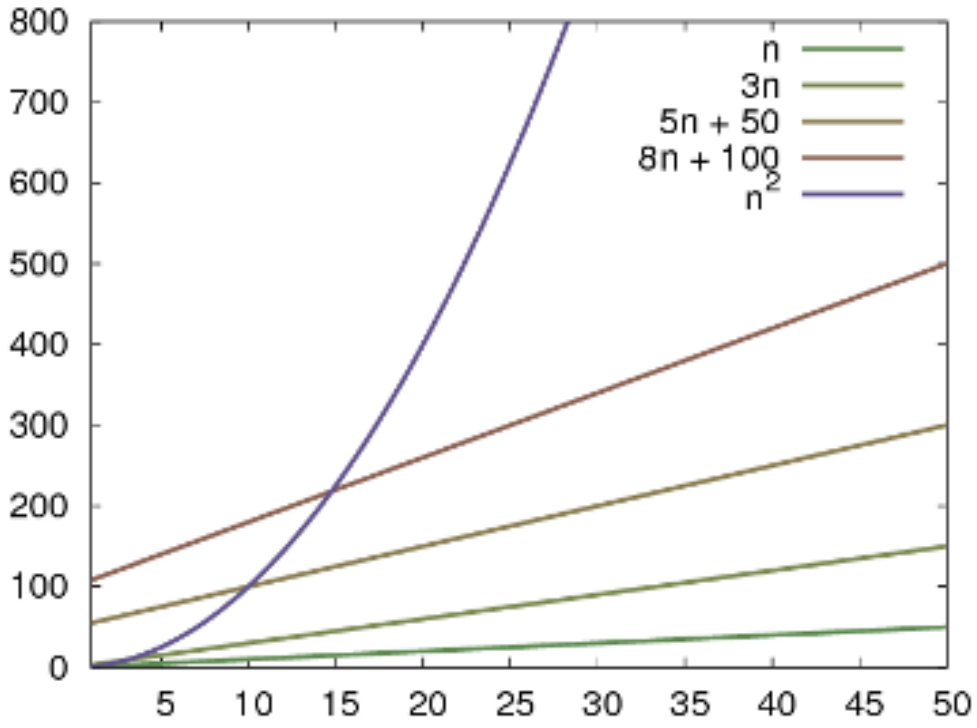




- Tera Web pages
- unbounded amount of Web logs
- billions of variables
- billions of transistors.

Happy Asymptotic World for Theoreticians

Efficient Algorithms for Big Data



Quadratic time algorithms could be too slow!!!!

Modern Notion of

Algorithmic Efficiency

Therefore, more than ever before, it is not just desirable, but essential, that efficient algorithms should be scalable. In other words, their complexity should be nearly linear or sub-linear with respect to the problem size.

Thus, scalability — not just polynomial-time computability — should be elevated as the central complexity notion for characterizing efficient computation.

Big Data and Scalable Algorithms

A Practical Match Made in the Digital Age

Big Data and Scalable Algorithms

$$\text{scalability}(A, x) = \frac{T_A(x)}{\text{size}(x)}$$

Big Data and Scalable Algorithms

$$\text{scalability}(A, x) = \frac{T_A(x)}{\text{size}(x)}$$

$$\text{scalability}_A(n) = O(\log^c n)$$

Big Data and Scalable Algorithms

$$\text{scalability}(A, x) = \frac{T_A(x)}{\text{size}(x)}$$

$$\text{scalability}_A(n) = O(\log^c n)$$

- Nearly-Linear Time Algorithms
- Sub-Linear Time Algorithms

Algorithmic Paradigms: Scorecard

- **Greedy** — **often scalable (limited applications)**
- **Dynamic Programming** — **usually not scalable (even when applicable)**
- **Divide-and-Conquer** — **sometimes scalable**
- **Mathematical Programming** — **rarely scalable**

- **Branch-and-Bound** — **hardly scalable**
- **Multilevel Methods** — **mostly scalable (lack of proofs)**
- **Local Search and Simulated Annealing** — **can be scalable**

Examples: Scalable Geometry Algorithms

Sorting O(n logn)

Nearest neighbors

Delaunay Triangulation/3D convex hull

Fixed Dimensional Linear Programming O(n)

ε–net in fixed-dimensional VC space

Examples: Scalable Graph Algorithms

Breadth-First Search O(|V|+|E|)

Depth-First Search

Shortest Path Tree

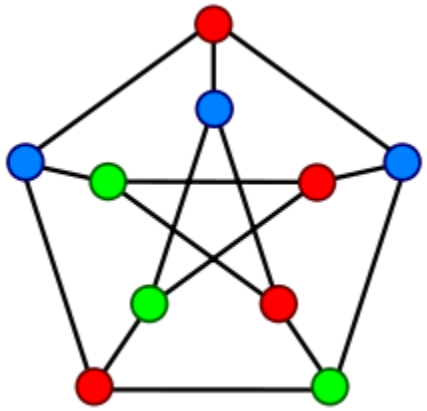
Minimum Spanning Tree

Planarity Testing

Bi-connected components

Topological sorting

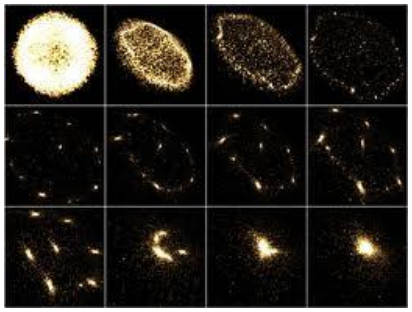
Sparse matrix vector product



Examples:
Scalable Numerical Algorithms

N-Body simulation

Sparse matrix vector product



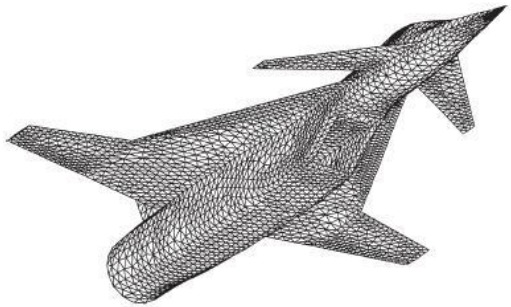
O(n)

FFT/Multiplication

O(n log n)

Multilevel algorithms

Multigrid

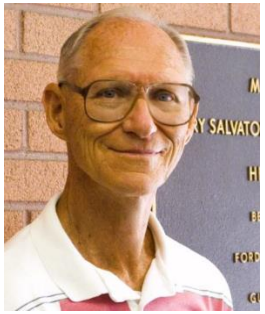


We need more provably-good scalable algorithms for network analysis, data mining, and machine learning (in real-time applications)

Scalable Methodology: Talk Outline

- **Scalable Primitives and Reduction**
  - **The Laplacian Paradigm**
    - Electrical Flows & Maximum Flows; Spectral Approximation; Tutte's embedding and Machine Learning
- **Scalable Technologies:**
  - **Spectral Graph Sparsification**
    - Sparse Newton's Method and Sampling from Graphical Models
  - **Computing Without the Whole Data: Local Exploration and Advanced Sampling**
    - Significant PageRanks
- ***Challenges: Computation over Dense/High-Dimensional Models with Succinct/Sparse Representations***
    - Social Influence; high order clustering;

Scalable Primitives and Reduction



Algorithm Design is like Building a Software Library

Scalable Reduction: Once scalable algorithms are developed, they can be used as *primitives or subroutines* for designing new scalable algorithms.

Laplacian Primitive

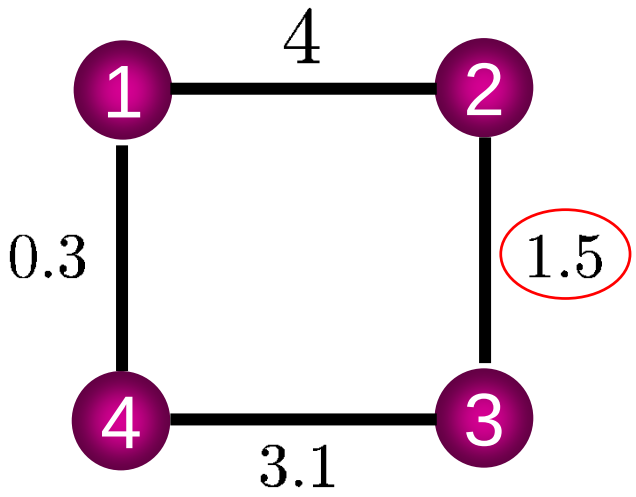
Solve $A\,x = b$, where A is a weighted Laplacian matrix

Laplacian Primitive

Solve $A\,x = b$, where A is a weighted Laplacian matrix

A is Laplacian matrix: symmetric non-positive off diagonal row sums = 0 Isomorphic to weighed graphs

$$\begin{pmatrix} 4.3 & -4 & 0 & -0.3 \\ -4 & 5.5 & -1.5 & 0 \\ 0 & -1.5 & 4.6 & -3.1 \\ -0.3 & 0 & -3.1 & 3.4 \end{pmatrix}$$



Scalable Laplacian Solvers (Spielman-Teng)

For symmetric diagonally dominant (SDD) ***A***, any ***b***

Compute $\left\|x - A^{-1}b\right\|_A < \epsilon \left\|x\right\|_A$ in time

$$m \log^{O(1)} n \log(1/\epsilon)$$

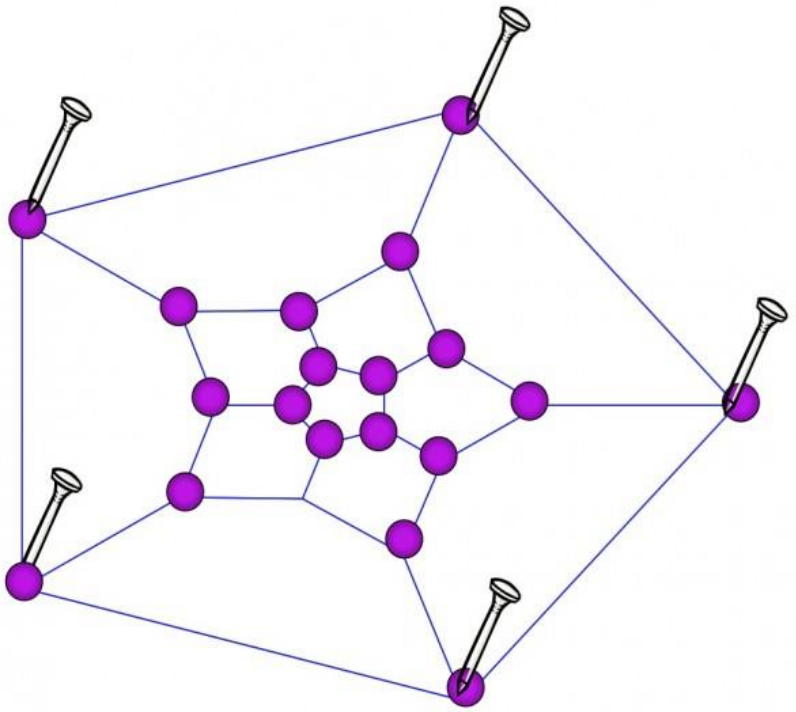
Greatly improved by Koutis-Miller-Peng, Kelner-Orecchia-Sidford-Zhu, …, Lee, Peng, and Spielman, to essentially O(m log (1/ε))

The Laplacian Paradigm

To apply the *Laplacian Paradigm* to solve a problem defined on massive networks or big matrices, we *attempt* to reduce the computational and optimization problem to one or more linear algebraic or spectral graph-theoretical problems.

Beyond scalable Laplacian solvers

Scalable Tutte's Embedding



Learning from labeled data on directed graphs [Zhou-Huang-Schölkopf]

Scalable Spectral Approximation

Approximate Fiedler Vector

For Laplacian ***A***, is vector $v^T \mathbf{1} = 0$ such that

$$\frac{v^T A v}{v^T v} \leq (1+\epsilon)\lambda_2(A)$$

Can find v using inverse power method, in time

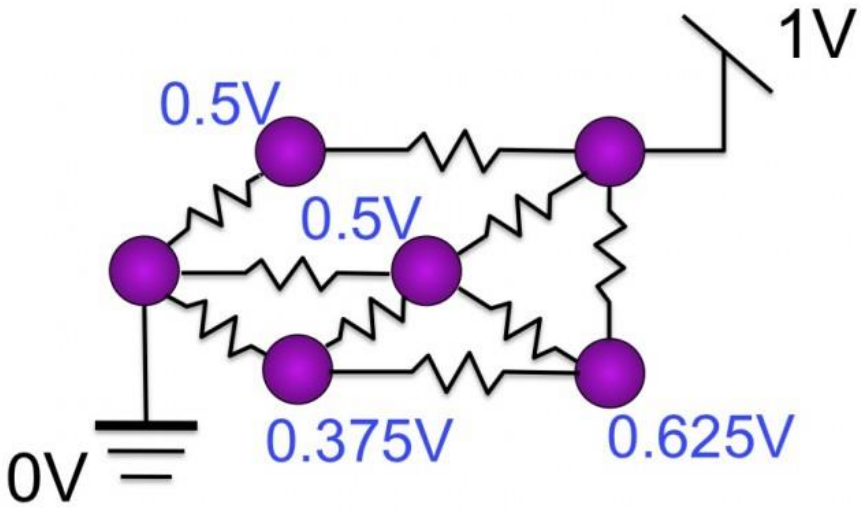
$$m \log^{O(1)} n \log(1/\epsilon)/\epsilon$$

Scalable Cheeger Cut

Theorem: Constant degree graph G, Fiedler value λ:
scalable computation of a cut of conductance

$$o\left(\sqrt{\lambda}\right)$$

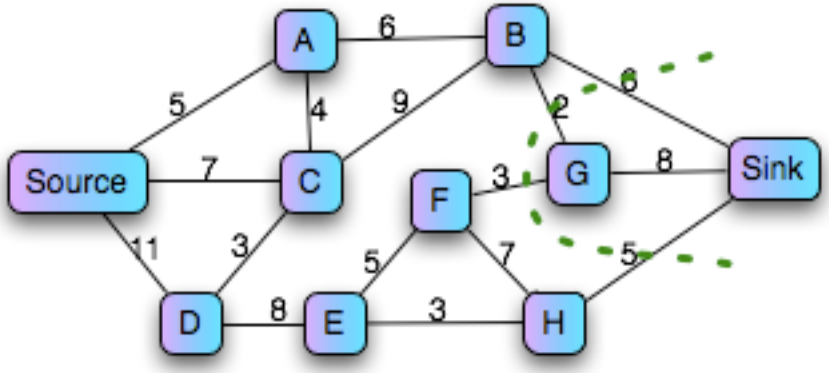
Scalable Electrical Flows



Electrical potentials: $L\,\varphi = \chi_{st}$

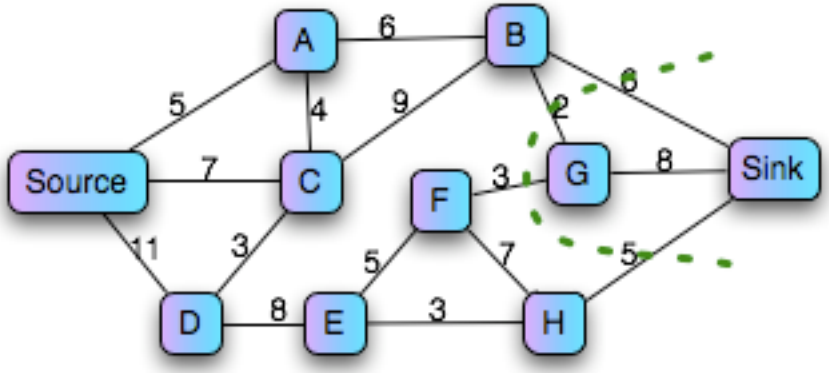
in time $\tilde{O}(m \log \varepsilon^{-1})$

Undirected Maximum Flow



Previously Best: O($m^{3/2}$) [Even-Tarjan 75]

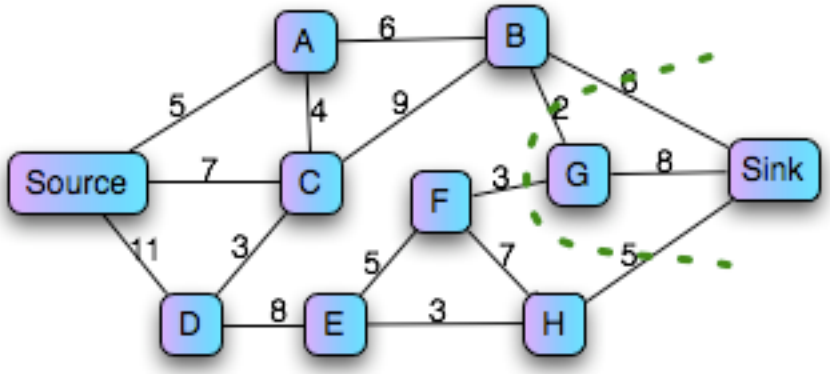
Maximum Flow
(Christiano-Kelner-Mądry-Spielman-Teng)



Iterative Electrical Flows: $\varphi = L^{-1} \chi_{st}$: $\tilde{O}(m^{4/3} \varepsilon^{-3})$

Previously Best: $\tilde{O}(\min(m^{3/2}, m\, n^{2/3}))$ [Goldberg-Rao]

Path to Scalable Maximum Flow



Previously Best: $\tilde{O}(\min(m^{3/2}, m\, n^{2/3}))$ [Goldberg-Rao]

Iterative Electrical Flows: $\varphi = L^{-1}\, \chi_{st}$: $\tilde{O}(m^{4/3}\, \varepsilon^{-3})$

Scalable: Sherman; Kelner-Lee-Orecchia-Sidford, **Peng**

Applications of The Laplacian Paradigm

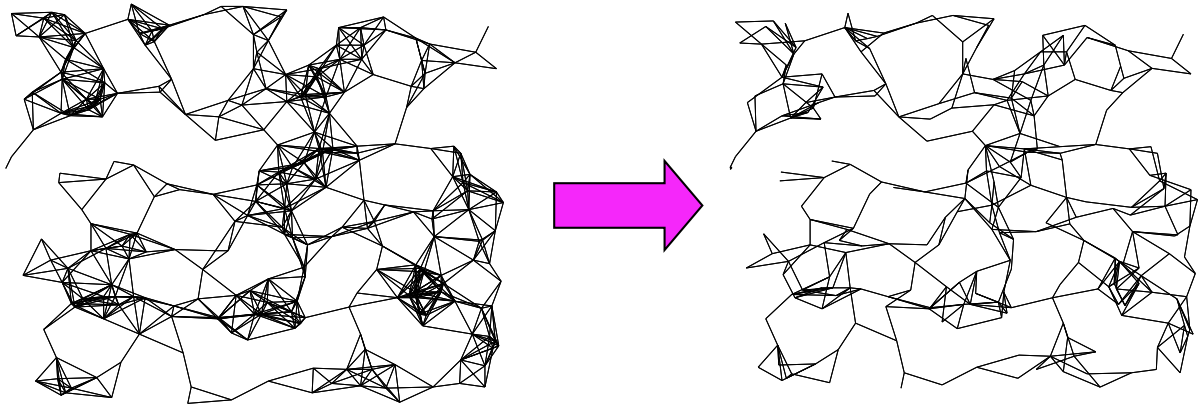
- Electrical flow computation
- Spectral approximation
- Tutte's embedding
- Learning from labeled data on a directed graph [Zhou-Huang-Schölkopf]
- Cover time approximation [Ding-Lee-Peres]
- Maximum flows and minimum cuts [Christiano-Kelner-Madry-Spielman-Teng]
- Elliptic finite-element solver [Boman-Hendrickson-Vavasis]
- Rigidity solver [Shklarski-Toledo; Daitch-Spielman]
- Image processing [Koutis-Miller-Tolliver]
- Effective resistances of weighted graphs [Spielman-Srivastava]
- Generation of random spanning trees [Madry-Kelner]
- Generalized lossy flows [Daitch-Spielman]
- Geometric means [Miller-Pachocki]
- …

Scalable Techniques

- **Algebraic Formulation of Network Problems**
- **Spectral Sparsification of Matrices and Networks**
- **Computing without the Whole Data: Local Exploration of Networks**

Graph Spectral Sparsifiers

For a graph G (with Laplacian L), a sparsifier is a graph $\tilde{G}$ (with Laplacian $\tilde{L}$) with at most $n \log^{O(1)} n$ edges s.t.

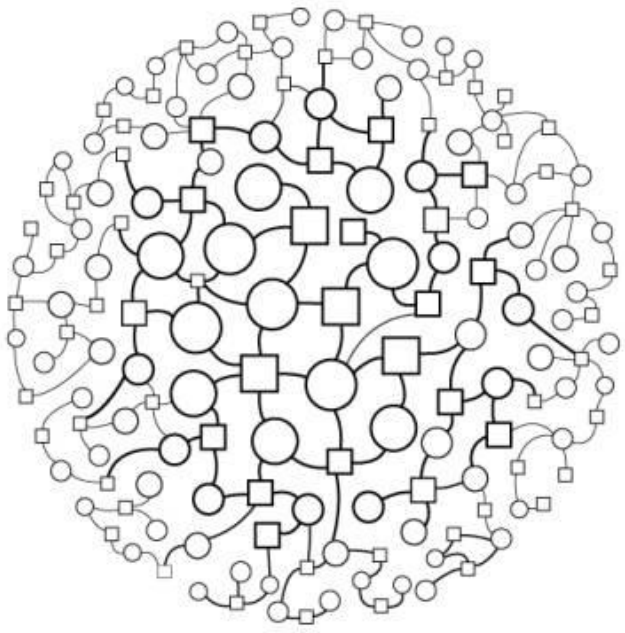


$$\kappa_f(L, \tilde{L}) \leq (1+\epsilon) \qquad \boxed{\forall x : x^T \tilde{L} x \leq x^T L x \leq (1+\epsilon) x^T \tilde{L} x}$$

Improved by Batson, Spielman, and Srivastava

Sampling From Graphical Models

Joint probability distribution of n-dimensional random variables **x**

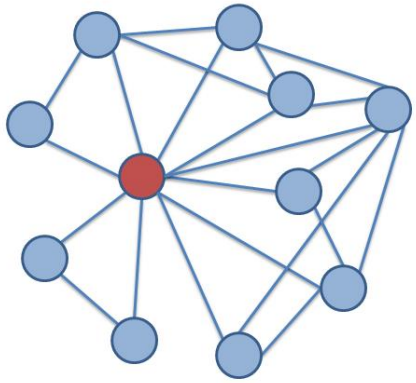


Graphical model for local dependencies

Sampling according to the model

Gibbs' Markov Chain Monte Carlo Process

Locally resample each variable, conditioned on the values of its graphical neighbors



- In limit, exact mean and covariance

  [Hammersley-Clifford]

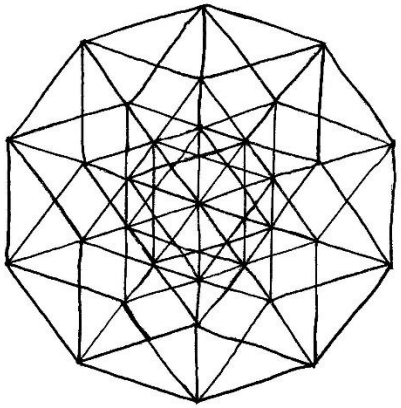
- Easy to implement

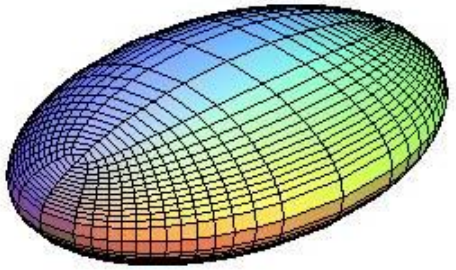
- Many iterations
- Sequential

A Holy Grail Sampling Question

Characterization of graphical models that have scalable parallel sampling algorithms with poly-logarithmic depth?

Gaussian Markov Random Fields





$$\Pr\left[\mathbf{x}|\mathbf{\Lambda}, \boldsymbol{h}\right] \propto \exp(-\frac{1}{2}\mathbf{x}^T\mathbf{\Lambda}\mathbf{x} + \boldsymbol{h}^T\mathbf{x})$$

- Precision matrix – symmetric positive definite
- Potential vector
- Goal: Sampling from Gaussian distribution $\mathbf{N}(\Lambda^{-1} h\ ,\ \mathbf{\Lambda}^{-1})$

GMRF with H-Precision Matrices

Johnson-Saunderson-Willsky (*NIPS* 2013)

DΛD is SDD

If the precision matrix **Λ** is (generalized) diagonally dominant, then Hogwild Gibbs distributed sampling process converges

Scalable Parallel Gaussian Sampling?

- Time complexity: $O(nnz(\Lambda))$
- Parallel complexity: $O(\log n)$
- Randomness complexity: n

It remains open even if the precision matrix is symmetric diagonally dominant (SDD).

A Numerical Program for Gaussian Sampling

1. **Find the mean:**

$$\mu = \Lambda^{-1} h$$

2. **Compute an inverse square-root factor:**

$$CC^{T} = \Lambda^{-1}$$

3. **Sampling:**

generate standard Gaussian variables z

$$x = C z + \mu$$

Canonical Inverse Square-Root

Canonical Inverse Square-Root

Focus on normalized Laplacian:

$$\mathbf{L} = \mathbf{D} - \mathbf{W} = \mathbf{D}^{1/2}(\mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2})\mathbf{D}^{1/2}$$

I-X

Newton's Method

$$(\mathbf{I}-\mathbf{X})^{-1}=\left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)\left(\mathbf{I}-\frac{3}{4}\mathbf{X}^2-\frac{1}{4}\mathbf{X}^3\right)^{-1}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)$$

Newton's Method

$$(\mathbf{I}-\mathbf{X})^{-1} = \left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)\left(\mathbf{I}-\frac{3}{4}\mathbf{X}^2-\frac{1}{4}\mathbf{X}^3\right)^{-1}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)$$

Newton's method uses dense matrix multiplications, even when the original matrix is sparse. This is particularly the case in network analysis, where input graphs are usually sparse. Although Newton's method may converge rapidly, which provides a numerical framework for designing not only sequential but also parallel algorithms, its intermediate computation could be prohibitively expensive for handling big data.

Sparse Newton's Method

$$(\mathbf{I}-\mathbf{X})^{-1} = \left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)\left(\mathbf{I}-\frac{3}{4}\mathbf{X}^2-\frac{1}{4}\mathbf{X}^3\right)^{-1}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)$$

Spectral Sparsification

Sparse Newton Chain

$$(\mathbf{I}-\mathbf{X})^{-1}=\left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)\left(\mathbf{I}-\frac{3}{4}\mathbf{X}^2-\frac{1}{4}\mathbf{X}^3\right)^{-1}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)$$

$$[\mathbf{X}_0,\mathbf{X}_1,\ldots,\mathbf{X}_{d-1}]$$

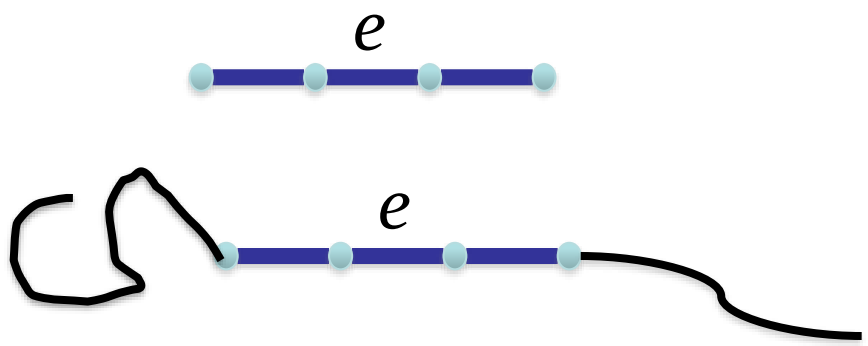
$\mathbf{X}_t$ is a spectral sparsifer of $\left(\frac{3}{4}\mathbf{X}_{t-1}^2+\frac{1}{4}\mathbf{X}_{t-1}^3\right)$

$$\mathbf{C}=\prod_{i=0}^{d-1}\left(\mathbf{I}+\frac{\mathbf{X}_i}{2}\right)$$

Random-Walk Polynomials and Sparsification

$$\mathbf{D} - \sum_{r=1}^{t} \alpha_r \mathbf{D} \cdot \left(\mathbf{D}^{-1}\mathbf{W}\right)^r$$

Path Sampling



Scalable Sparsification of Random-Walk Polynomials

$$O(t^2 \cdot m \cdot \log^2 n \cdot \frac{1}{\epsilon^2})$$

Scalable Parallel Gaussian Sampling for H-Precision Matrices

Cheng-Cheng-Liu-Peng-Teng (*COLT* 2015)

- Time complexity: $O(nnz(\Lambda))$
- Parallel complexity: $O(\text{polylog } n)$
- Randomness complexity: $2n$

Scalable Sparse Newton's Method

Matrix p^{th}-Power Factorization:

Given an $n \times n$ Laplacian matrix $\mathbf{M}$ and a constant $-1 \leq p \leq 1$, compute an $n \times n$ linear operator $\mathbf{C}$ such that $\mathbf{M}^p = \mathbf{C}\mathbf{C}^\top$.

$$(\mathbf{I}-\mathbf{X})^{-\frac{1}{q}} = \left(\mathbf{I}+\frac{1}{2q}\mathbf{X}\right)\left[\left(\mathbf{I}+\frac{1}{2q}\mathbf{X}\right)^{2q}(\mathbf{I}-\mathbf{X})\right]^{-\frac{1}{q}}\left(\mathbf{I}+\frac{1}{2q}\mathbf{X}\right)$$

Scalable Matrix Roots

Matrix p^{th}-Power Factorization:
Given an $n \times n$ Laplacian matrix $\mathbf{M}$ and a constant $-1 \leq p \leq 1$, compute an $n \times n$ linear operator $\mathbf{C}$ such that $\mathbf{M}^p = \mathbf{C}\mathbf{C}^\top$.

Nick Higham at Brain Davies' 65 Birthday: An email from a power company regarding the usage of electricity networks

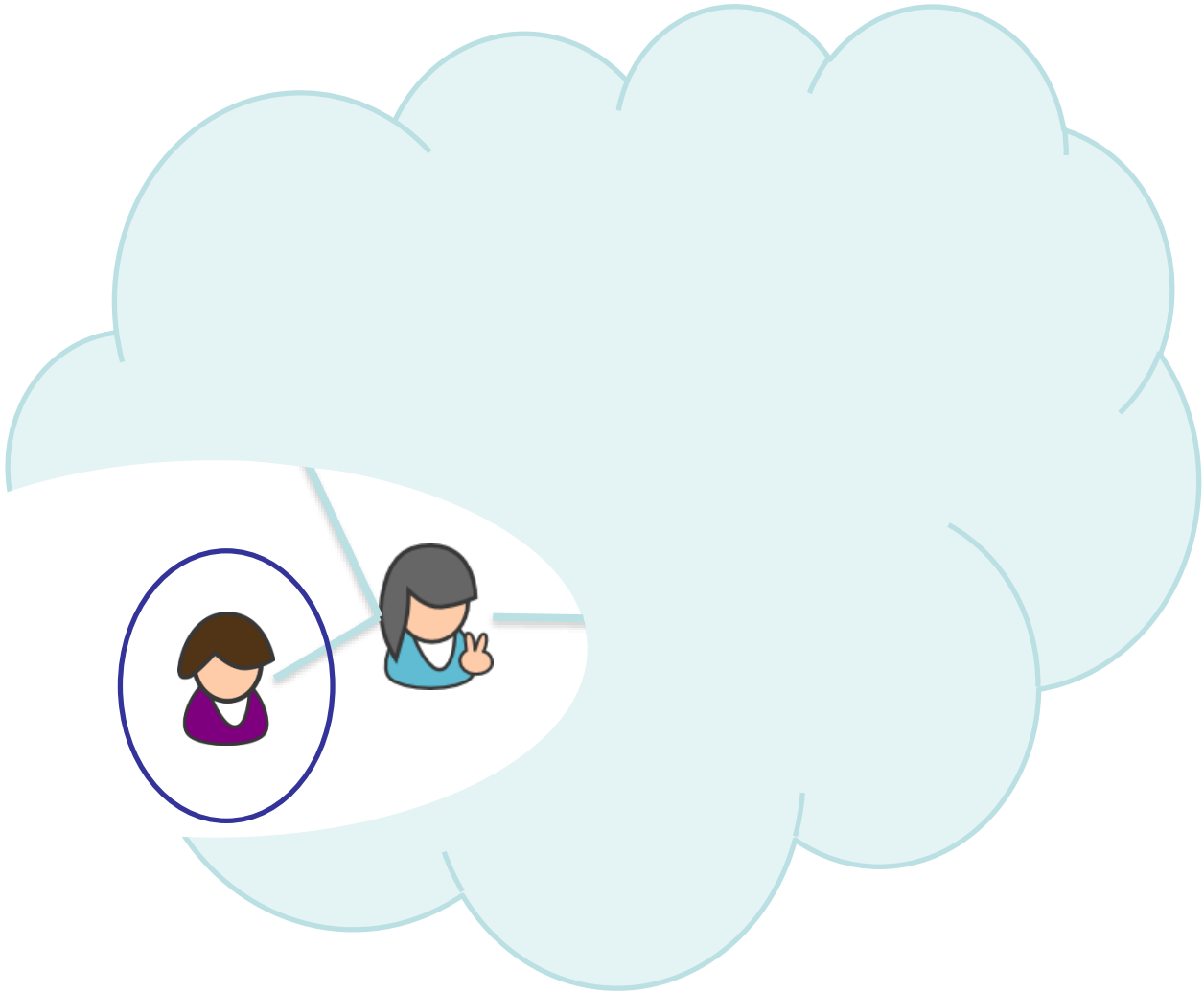
"*I have an Excel spreadsheet containing the transition matrix of how a company's [Standard & Poor's] credit rating charges from on year to the next. I'd like to be working in eighths of a year, so the aim is to find the eighth root of the matrix.*"

Family of Scalable Techniques

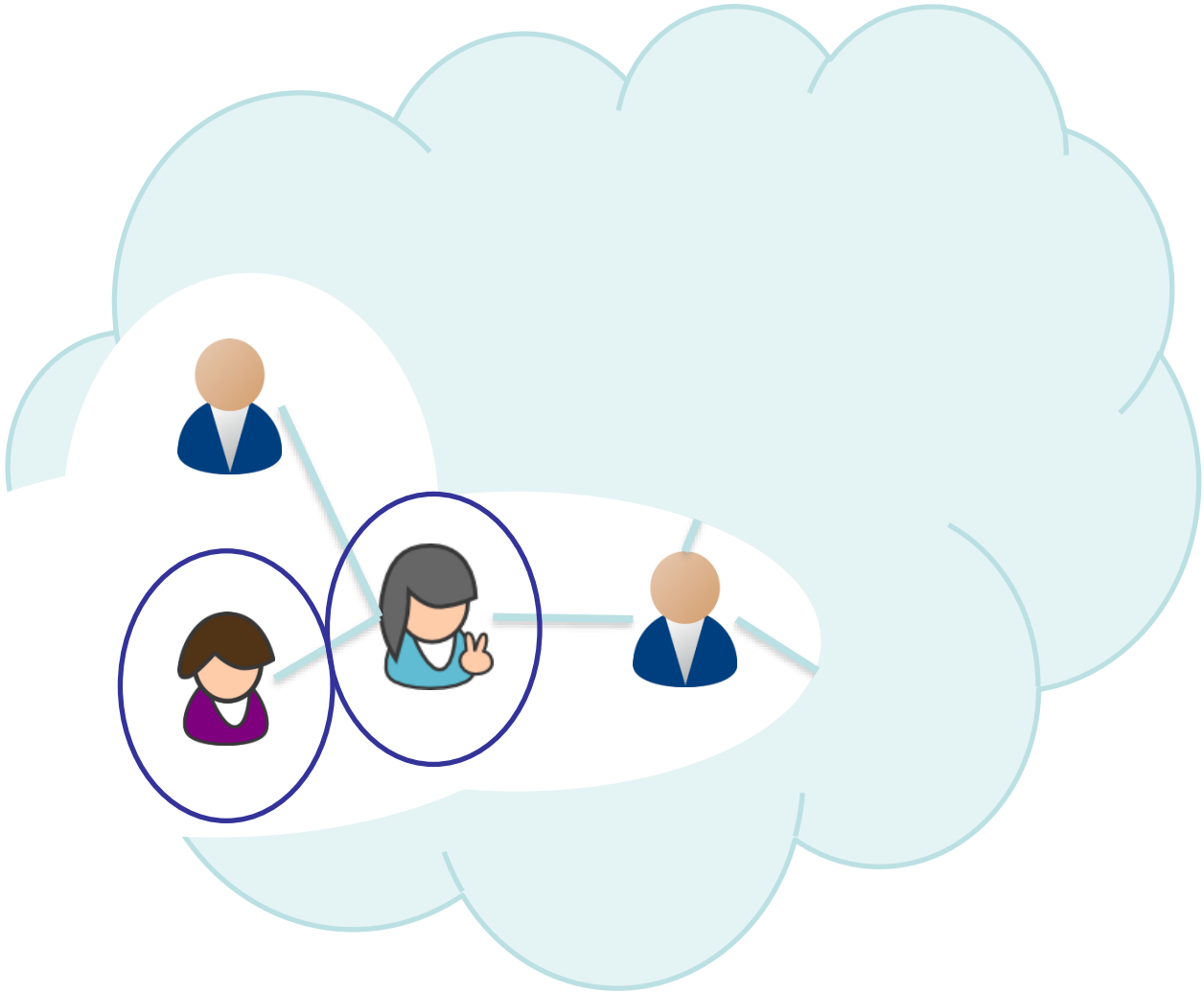
- **Algebraic Formulation of Network Problems**
- **Spectral Sparsification of Matrices and Networks**
- **Computing without the Whole Data: Local Exploration of Networks**

Local Network Algorithms

Local Network Algorithms



Local Network Algorithms

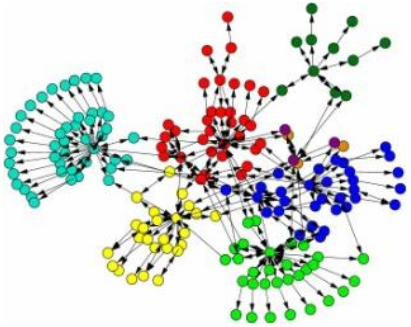


Local Network Algorithms

Local Network Algorithms

PageRank

- **PageRank:** Stationary Distribution of the Markov Process:



  - Probability 1–α: random walk on the network
  - Probability α: random restarting

  - Stationary Distribution:

$$\mathbf{PR}_{\mathbf{W},\alpha} = \alpha \cdot \mathbf{1} + (1-\alpha) \cdot \mathbf{PR}_{\mathbf{W},\alpha} \cdot \left(\mathbf{D}_{\mathbf{W}}^{out}\right)^{-1} \mathbf{W}$$

Significant PageRank without Explore the Entire Network?

Input: G, $1 \leq \Delta \leq n$, and $c>1$

Output: Identify a subset $S \subseteq V$ containing:

- all nodes of PageRank at least Δ
- no nodes with PageRank less than Δ/c

O(n/Δ) time algorithm?

Personalized PageRank Matrix

Personalized PageRank

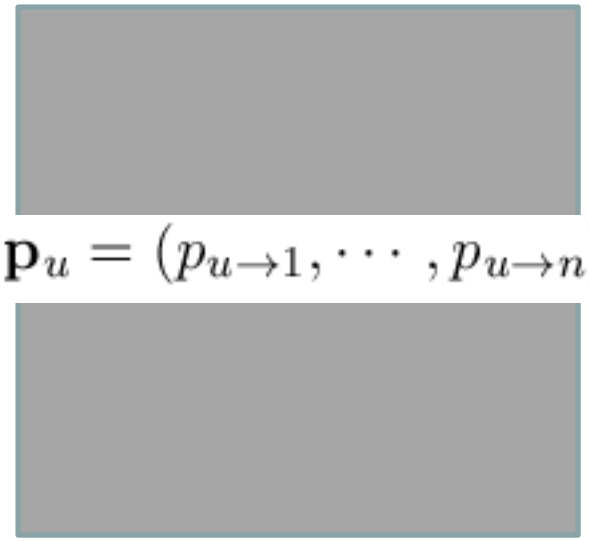
$$\mathbf{p}_u = \alpha \cdot \mathbf{1}_u + (1-\alpha) \cdot \mathbf{p}_u \cdot \left(\mathbf{D}_{\mathbf{W}}^{out}\right)^{-1} \cdot \mathbf{W}$$

$$\mathbf{p}_u = (p_{u\to 1}, \cdots, p_{u\to n})$$

$$\mathbf{PPR}_{\mathbf{W},\alpha} = [\mathbf{p}_1, ..., \mathbf{p}_n]^T = \begin{bmatrix} p_{1\to 1} & \cdots & p_{1\to n} \\ \vdots & \cdots & \vdots \\ p_{n\to 1} & \cdots & p_{n\to n} \end{bmatrix}$$

Scalable Local Personalized PageRank

$$\mathbf{p}_u = \alpha \cdot \mathbf{1}_u + (1-\alpha) \cdot \mathbf{p}_u \cdot \left(\mathbf{D}_{\mathbf{W}}^{out}\right)^{-1} \cdot \mathbf{W}$$



Jed-Widom

Andersen-Chung-Lang

$O(d_{max}/\varepsilon)$

Fogaras-Racz-Csalogany-Sarlos

Borgs-Brautbar-Chayes-Teng

$O(\log n/(\varepsilon\rho^2))$

An Abstract Problem: Vector Sum

Input: $\boldsymbol{v}$ (an unknown vector from $[0,1]^n$)

$1 \leq \Delta \leq n$ (a threshold value)

Query Model: $?(\boldsymbol{v},i,\varepsilon)$

Cost: $1/\varepsilon$

Output: Is sum($\boldsymbol{v}$) more than Δ or less than $\Delta/2$?

Question: ***O*(*n*/Δ) *cost algorithm?***

Riemann Estimator

Borgs-Brautbar-Chayes-Teng

$$S_Q = \frac{n}{T}\sum_{t=1}^{T}\mathbf{I}[q_t \geq \epsilon_t],\ \text{where}\ \forall t \in [T],\ \epsilon_t = \frac{t}{T}$$

$$\mathrm{E}[q] = \int_0^1 \Pr[q \geq x]\, dx$$

$$\tilde{O}\left(\frac{n}{\Delta}\right)$$

Scalable Methodology: Talk Outline

- **Scalable Primitives and Reduction**
  - **The Laplacian Paradigm**
    - Electrical Flows & Maximum Flows; Spectral Approximation; Tutte's embedding and Machine Learning
- **Scalable Technologies:**
  - **Spectral Graph Sparsification**
    - Sparse Newton's Method and Sampling from Graphical Models
  - **Computing Without the Whole Data: Local Exploration and Advanced Sampling**
    - Significant PageRanks

Challenges

- **Computation over dense models with succinct/sparse representations**
  - High order clustering;
- **Computation over high-dimensional models with succinct/sparse representations**
  - Social Influence;
- **Computation over incomplete data**
  - ML

Clustering Based on Personalized Page-Rank Matrix

$$\begin{bmatrix} p_{1\to 1} & \cdots & p_{1\to n} \\ \vdots & \cdots & \vdots \\ p_{n\to 1} & \cdots & p_{n\to n} \end{bmatrix}$$

$$\text{PPR-Density}_{\mathbf{w},\alpha}(S) = \frac{1}{|S|} \cdot \sum_{u,v\in S} \mathbf{PPR}_{\mathbf{w},\alpha}[u,v]$$

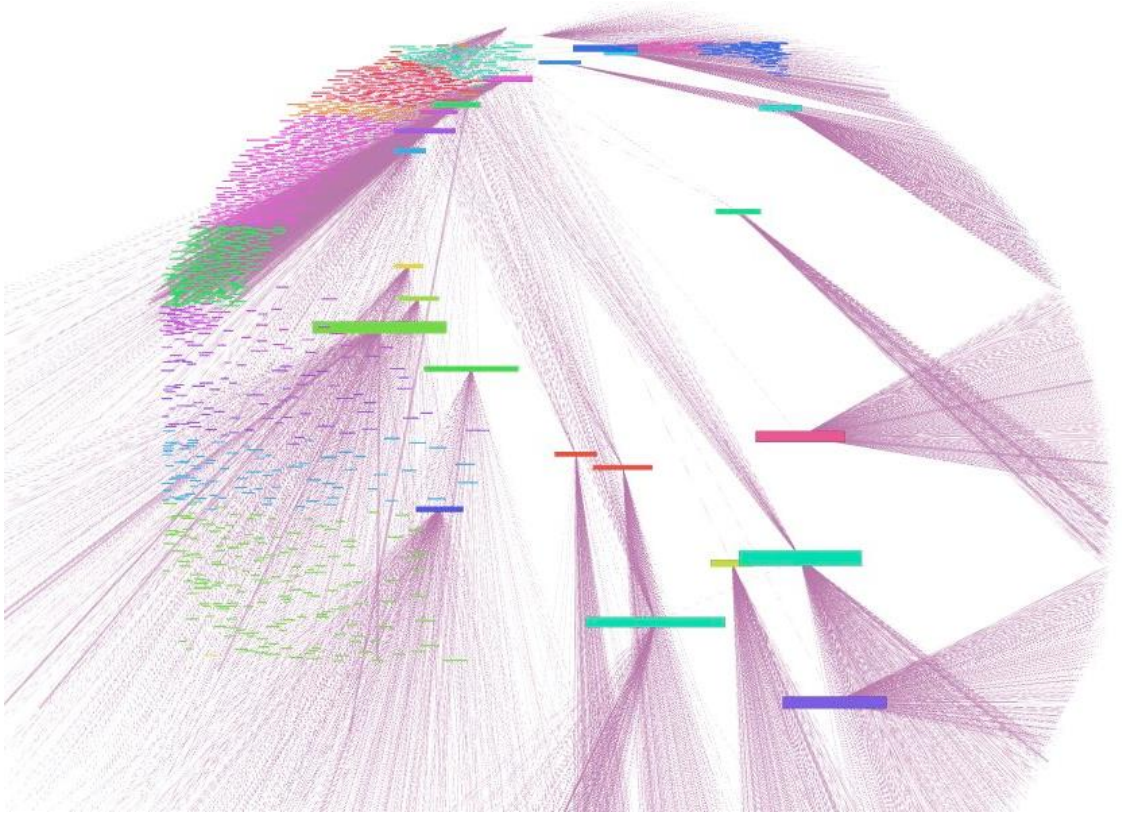
suspiciousness measure (Hooi-Song-Beutel-Shah-Shin-Faloutsos)

Open Question: scalable 2-approximation?

Reversed Diffusion Structure and Process

$$\begin{bmatrix} p_{1\to 1} & \cdots & p_{1\to n} \\ \vdots & \cdots & \vdots \\ p_{n\to 1} & \cdots & p_{n\to n} \end{bmatrix}$$

Influence Through Social Networks



Reversed Diffusion Process

- **Scalable Influence Maximization**
  - Borgs, Brautbar, Chayes, Lucier
  - Tang, Shi, Xiao
- **Scalable Shapley Centrality of Social Influence**
  - Chen and Teng

Local Graph Clustering
Spielman-Teng

Given a vertex *v* of interest in a massive network

find a provably-good cluster near *v*

in time *O*(cluster size)

Open Question: What other clustering problems can it be extended to?
High order clustering?

Foundations and Trends® in Theoretical Computer Science > Vol 12 > Issue 1–2

Scalable Algorithms for Data and Network Analysis

Shang-Hua Teng, University of Southern California, Los Angeles, USA, shanghua@usc.edu

Suggested Citation
Shang-Hua Teng (2016), "Scalable Algorithms for Data and Network Analysis", Foundations and Trends® in Theoretical Computer Science: Vol. 12: No. 1–2, pp 1-274. http://dx.doi.org/10.1561/0400000051 Export

Published: 30 May 2016

Subjects
Algorithmic game theory, Computational aspects of combinatorics and graph theory, Computational complexity, Computational geometry, Computational Models and Complexity, Data structures, Design and analysis of algorithms, Operations Research, Parallel algorithms, Randomness in Computation, Data Mining, Economics of information and the Web, Scalability, Social Networking, Spectral methods, Robustness, Optimization, Markov chain Monte Carlo, Graphical models, Game theoretic learning, Dimensionality reduction, Data mining, Clustering, Web search, Natural language processing for IR

Journal details

Abstract

In the age of Big Data, efficient algorithms are now in higher demand more than ever before. While Big Data takes us into the asymptotic world envisioned by our pioneers, it also challenges the classical notion of efficient algorithms: Algorithms that used to be considered efficient, according to polynomial-time characterization, may no longer be adequate for solving today's problems. It is not just desirable, but essential, that efficient algorithms should be scalable. In other words, their complexity should be nearly linear or sub-linear with respect to the problem size. Thus, scalability, not just polynomial-time computability, should be elevated as the central complexity notion for characterizing efficient computation. In this tutorial, I will survey a family of algorithmic techniques for the design of provably-good scalable algorithms. These techniques include local network exploration, advanced sampling, sparsification, and geometric partitioning. They also include spectral graph-theoretical methods, such as those used for computing electrical flows and sampling from Gaussian Markov random fields. These methods exemplify the fusion of combinatorial, numerical, and statistical thinking in network analysis. I will illustrate the use of these techniques by a few basic problems that are fundamental in network analysis, particularly for the identification of significant nodes and coherent clusters/communities in social and information networks. I also take this opportunity to discuss some frameworks beyond graph-theoretical models for studying conceptual questions to understand multifaceted network data that arise in social influence, network dynamics, and Internet economics.

DOI:10.1561/0400000051

Foundations and Trends® in
sample
Vol. xx, No xx (xxxx) 1–211
© xxxx xxxxxxxxx
DOI: xxxxxx

now
the essence of knowledge

Scalable Algorithms for Data and Network Analysis[1]

Shang-Hua Teng[1]

[1] *Computer Science, USC. Los Angeles, CA 90089, USA, shanghua@usc.edu*

Abstract

In the age of Big Data, efficient algorithms are now in higher demand more than ever before. While Big Data takes us into the asymptotic world envisioned by our pioneers, it also challenges the classical notion of efficient algorithms: Algorithms that used to be considered efficient, according to polynomial-time characterization, may no longer be adequate for solving today's problems. It is not just desirable, but essential, that efficient algorithms should be *scalable*. In other words, their complexity should be nearly linear or sub-linear with respect to the problem size. Thus, *scalability*, not just polynomial-time computability, should be elevated as the central complexity notion for characterizing efficient computation.

In this article, I will survey a family of algorithmic techniques

Big Data and Network Sciences: Going Beyond Graph Theory

- **Set Functions**
- **Distributions**
- **Dynamics**
- **Multilayer Networks**

Thanks

- **Coauthors:** Dan Spielman; Paul Christiano, Jon Kelner, Aleksander Mądry, Dehua Cheng, Yu Cheng, Yan Liu, Richard Peng, Christian Borgs, Michael Brautbar, Jennifer Chayes, Wei Chen

- **Funding**: NSF (*large*) and Simons Foundation (*curiosity-driven investigator award*)

Scalable Algorithms for Big Data and Network Sciences: Characterization, Primitives, and Techniques

Shang-Hua Teng

USC