Algorithmic Primitives for Network Analysis: Through the Lens of the Laplacian Paradigm

Shang-Hua Teng
Computer Science, Viterbi School of Engineering
USC
Massive Data and Massive Graphs

- 500 billions web pages
- unbounded amount of web logs
- billions of variables
- billions of transistors.

Happy Asymptotic World for Theoreticians
Efficient Algorithm Challenges

Nearly-Linear-Time Algorithms

Quadratic time algorithms could be too slow!!!!
Algorithmic Paradigms

• Greedy          often nearly linear (limited applications)
• Dynamic Programming usually not linear (even when applicable)
• Divide-and-Conquer   sometimes nearly linear
• Mathematical Programming rarely linear

• Branch-and-Bound        hardly linear
• Multilevel Methods       mostly linear (lack of proofs)
• Local Search and Simulated Annealing can be nearly linear
Examples: Nearly-Linear-Time Graph Algorithms

Breadth-First Search
Depth-First Search
Shortest Path Tree
Minimum Spanning Tree
Planarity Testing
Bi-connected components
Topological sorting
Sparse matrix vector product

$O(|V|+|E|)$
Algorithm Design is like Building a Software Library

Once a nearly-linear time (sub-linear time) algorithm is developed, it can be used as a primitive or a subroutine in the design of other nearly-linear-time (sub-linear time) 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

$$
\begin{pmatrix}
4.3 & -4 & 0 & -0.3 \\
-4 & 5.5 & -1.5 & 0 \\
0 & -1.5 & 4.6 & -3.1 \\
-0.3 & 0 & -3.1 & 3.4
\end{pmatrix}
$$
Nearly Linear-Time Laplacian Solver
(Spielman-Teng)

For symmetric, diagonally dominant $A$, any $b$

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

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

Improved by [Koutis-Miller-Peng] to essentially $O(m \log m \log (1/\epsilon))$
The Laplacian Paradigm

To apply the Laplacian Paradigm to solve a problem defined on a massive graph or a matrix, we reduce the computational and optimization problem to one or more linear algebraic or spectral graph-theoretic problems whose matrices are Laplacian or Laplacian-like.

We then apply a nearly linear-time Laplacian solver
Electrical Flows
Electrical Flows
(Kelner-Spielman-Teng)

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

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

Input: directed graph $G$ with integer capacities $u(\cdot)$, source $s$ and sink $t$

Task: Find a feasible $s$-$t$ flow of max value
Maximum Flow: A Classic and Fundamental Optimization Problem

- Extensively studied since 1930s
- Broadly applied in practice
- Linear programming duality
  - maxflow and mincut
- Extremely influential in development of theory
  - (classical ‘textbook problem’)

[Diagram of network flow]
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]
Spectral Approximation

Approximate Fiedler Vector

For Laplacian $A$, is vector $v^T I = 0$ such that

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

Can find $v$ using inverse power method, in time

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

**Theorem:** If $G$ is a constant degree graph of $n$ nodes with Fiedler value $\lambda$, then in nearly-linear-time, we can compute a cut of conductance $O(\sqrt{\lambda})$.

**Corollary:** If $G$ is a bounded-degree graph with a bounded genus, then in nearly-linear-time one can compute a cut of conductance $O(\sqrt{1/n})$, without any pre-computation of the embedding.
Applications of The Laplacian Paradigm

- Spectral approximation [Spielman-Teng]
- Electrical flow computation [Kelner-Spielman-Teng]
- Maximum flows and minimum cuts [Christiano-Kelner-Madry-Spielman-Teng]
- Cover time approximation [Ding-Lee-Peres]

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

...
Applications of The Laplacian Paradigm

- Spectral approximation [Spielman-Teng]
- Electrical flow computation [Kelner-Spielman-Teng]
- Maximum flows and minimum cuts [Christiano-Kelner-Madry-Spielman-Teng]
- Cover time approximation [Ding-Lee-Peres]

- Learning from labeled data on a directed graph [Zhou-Huang-Schölkopf]
- 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]
- …
Talk Outline

• A New Primitive: The Laplacian Primitive

• The Laplacian Paradigm
  – Electrical Flows & Maximum Flows
  – Spectral Approximation
  – Machine Learning
  – Random Walks & Spanning Trees

• Related Algorithmic Primitives for Network Analysis
Clustering -- Graph Partitioning

\[ V = \bigcup_i V_i \]
Cluster = a subset of vertices

Conductance of $S$

$$\Phi(S) \overset{\text{def}}{=} \frac{\# \text{ edges leaving } S}{\text{sum of degrees on smaller side}}$$

Conductance of $G$

$$\Phi_G \overset{\text{def}}{=} \min_S \Phi(S)$$
Graph Partitioning

- Rounding Mathematical Programming: Too Slow
- Spectral Partitioning: Unbalanced
- Separator Theorems: Special Graphs
- Fast Heuristics: No Proof

- Nearly Linear-Time Partitioning?
Local Graph Clustering

Given a vertex $v$ of interest in a massive graph

find a cluster near $v$ with small conductance

in time $O(\text{cluster size})$

How should one explore from a starting vertex $v$?

Problem: given a vertex $v$,

find a cluster $S$ containing $v$, $\Phi(S)$ small,
in time proportional to size of $S$
Theorem [Statistical Guarantee]:
If $S$ is set of conductance $< c\phi^2 / \log^3 m$
$v$ is random vertex of $S$
Then output set $C$ of conductance $< \phi$
mostly in $S$,
in time proportional to $\deg(C')$

Main Technique: rounded random walk

Improved by Andersen-Chung-Lang; Andersen Peres
A Local-Clustering Algorithm

Algorithm **Nibble** (start vertex v, target set size k)

start: \( p^0(v) = 1 \)

step: \( p^{t+1} = p^t M \)

round all values \(< 1/k \log^{O(1)} n\) round to 0
Local Clustering Applications

Protein Network Analysis
  Networks specified by sets of protein-protein interactions
  Develop tool to locally explore protein networks
  Validate clusters using functional annotation
Local Communities in Protein Networks

http://gaussian.bu.edu/lpcf.html

[Choos your network:]

[Choose the starting protein:]

[Choose the community size:]

[Voevodski-Teng-Xia]
From Local Clustering to Nearly-Linear-Time Partitioning

If $G$ has a subset $S$ with $\text{Balance}(S) = b$, and

$$\Phi_{G'(S)} > c\phi^2 / \log^3 m$$

then, in nearly-linear time, **Approx-Cut** outputs $C$ with

$\text{Balance}(C) > b/2$, and

$$\Phi(C) < \phi$$

Improved by Andersen-Chung-Lang; Andersen Peres
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.

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

Improved by Batson, Spielman, and Srivastava
Exanpder is a Sparsifier of Complete Graph

If $G$ is complete graph on $n$ vertices,

$$x^T L_G x = n$$

If $H$ is $d$-regular Ramanujan expander,

$$\forall x \perp 1, \|x\| = 1$$

$$d - 2\sqrt{d - 1} \leq x^T L_H x \leq d + 2\sqrt{d - 1}$$

And so

$$\frac{n}{d + 2\sqrt{d - 1}} \cdot H \preceq G \preceq \frac{n}{d - 2\sqrt{d - 1}} \cdot H$$
**Sublinear Algorithm for Significant PageRanks**

Given a network $G = (V,E)$, a threshold value $1 \leq \Delta \leq n$, identify, with success probability $1-o(1)$, a subset $S \subseteq V$ with the property that $S$ contains all

- vertices of PageRank at least $\Delta$
- no vertex with PageRank less than $\Delta/2$

$O(n/ \Delta)$ time algorithm [Borgs-Brautbar-Chayes-Teng]

Multi-scale sampling of personalized Page Rank Matrix

$\varepsilon$-precision in $O(\log n/ \varepsilon)$ time
Endogenously Formed Communities of Affinity (Preference) Systems

Self-Certified Communities: [Balcan-Borgs-Braveman-Chayes-Teng]
Algorithmic Primitives for Network Analysis: Through the Lens of the Laplacian Paradigm

Shang-Hua Teng
Computer Science, Viterbi School of Engineering
USC

Thank you!