Approximate computation and implicit regularization for very large-scale data analysis

Michael W. Mahoney

Stanford University

May 2012

(For more info, see: http://cs.stanford.edu/people/mmahoney)
Algorithmic vs. Statistical Perspectives

Lambert (2000); Mahoney "Algorithmic and Statistical Perspectives on Large-Scale Data Analysis" (2010)

**Computer Scientists**
- **Data**: are a record of everything that happened.
- **Goal**: process the data to find interesting patterns and associations.
- **Methodology**: Develop approximation algorithms under different models of data access since the goal is typically computationally hard.

**Statisticians (and Natural Scientists, etc)**
- **Data**: are a particular random instantiation of an underlying process describing unobserved patterns in the world.
- **Goal**: is to extract information about the world from noisy data.
- **Methodology**: Make inferences (perhaps about unseen events) by positing a model that describes the random variability of the data around the deterministic model.
Perspectives are NOT incompatible

- Statistical/probabilistic ideas are central to recent work on developing improved randomized algorithms for matrix problems.

- Intractable optimization problems on graphs/networks yield to approximation when assumptions are made about network participants.

- In boosting (a statistical technique that fits an additive model by minimizing an objective function with a method such as gradient descent), the computation parameter (i.e., the number of iterations) also serves as a regularization parameter.
But they are VERY different paradigms

**Statistics, natural sciences, scientific computing, etc:**
- Problems often involve computation, but the study of computation *per se is secondary*
- Only makes sense to develop algorithms for well-posed* problems
- First, write down a model, and think about computation later

**Computer science:**
- Easier to study computation *per se in discrete settings*, e.g., Turing machines, logic, complexity classes
- Theory of algorithms divorces computation from data
- First, run a fast algorithm, and ask what it means later

*Solution exists, is unique, and varies continuously with input data*
How do we view BIG data?
Anecdote 1: Randomized Matrix Algorithms

Theoretical origins
- theoretical computer science, convex analysis, etc.
- Johnson-Lindenstrauss
- Additive-error algs
- Good worst-case analysis
- No statistical analysis

Practical applications
- NLA, ML, statistics, data analysis, genetics, etc
- Fast JL transform
- Relative-error algs
- Numerically-stable algs
- Good statistical properties

How to "bridge the gap"?
- decouple randomization from linear algebra
- importance of statistical leverage scores!

Mahoney "Algorithmic and Statistical Perspectives on Large-Scale Data Analysis" (2010)
Mahoney "Randomized Algorithms for Matrices and Data" (2011)
Anecdote 2: Communities in large informatics graphs

People imagine social networks to look like:

Real social networks actually look like:

How do we know this plot is “correct”?

• (since computing conductance is intractable)
• Algorithmic Result (ensemble of sets returned by different approximation algorithms are very different)
• Statistical Result (Spectral provides more meaningful communities than flow)
• Lower Bound Result; Structural Result; Modeling Result; Etc.

Data are expander-like at large size scales !!!

There do not exist good large clusters in these graphs !!!
Lessons from the anecdotes

We are being forced to engineer a union between two very different worldviews on what are fruitful ways to view the data

• in spite of our best efforts not to

Often fruitful to consider the statistical properties implicit in worst-case algorithms

• rather that first doing statistical modeling and then doing applying a computational procedure as a black box

• for both anecdotes, this was essential for leading to “useful theory”

How to extend these ideas to “bridge the gap” b/w the theory and practice of MMDS (Modern Massive Data Set) analysis.

• QUESTION: Can we identify a/the concept at the heart of the algorithmic-statistical disconnect and then drill-down on it?
Outline and overview

Preamble: algorithmic & statistical perspectives

General thoughts: data, algorithms, and explicit & implicit regularization

Approximate first nontrivial eigenvector of Laplacian
• Three random-walk-based procedures (heat kernel, PageRank, truncated lazy random walk) are implicitly solving a regularized optimization exactly!

Spectral versus flow-based algs for graph partitioning
• Theory says each regularizes in different ways; empirical results agree!

Weakly-local and strongly-local graph partitioning methods
• Operationally like L1-regularization and already used in practice!
Outline and overview

Preamble: algorithmic & statistical perspectives

**General thoughts:** data, algorithms, and explicit & implicit regularization

Approximate first nontrivial eigenvector of Laplacian

• Three random-walk-based procedures (heat kernel, PageRank, truncated lazy random walk) are *implicitly* solving a regularized optimization *exactly*!

Spectral versus flow-based algs for graph partitioning

• Theory says each regularizes in different ways; empirical results agree!

Weakly-local and strongly-local graph partitioning methods

• Operationally like L1-regularization and already used in practice!
Thoughts on models of data (1 of 2)

Data are whatever data are

- records of banking/financial transactions, hyperspectral medical/astronomical images, electromagnetic signals in remote sensing applications, DNA microarray/SNP measurements, term-document data, search engine query/click logs, user interactions on social networks, corpora of images, sounds, videos, etc.

To do something useful, you must model the data

Two criteria when choosing a data model

- (data acquisition/generation side): want a structure that is “close enough” to the data that you don’t do too much “damage” to the data

- (downstream/analysis side): want a structure that is at a “sweet spot” between descriptive flexibility and algorithmic tractability
Thoughts on models of data (2 of 2)

Examples of data models:

• *Flat tables and the relational model:* one or more two-dimensional arrays of data elements, where different arrays can be related by predicate logic and set theory.

• *Graphs, including trees and expanders:* \( G = (V,E) \), with a set of nodes \( V \) that represent “entities” and edges \( E \) that represent “interactions” between pairs of entities.

• *Matrices, including SPSD matrices:* \( m \) “objects,” each of which is described by \( n \) “features,” i.e., an \( n \)-dimensional Euclidean vector, gives an \( m \times n \) matrix \( A \).

*Much modern data are relatively-unstructured; matrices and graphs are often useful, especially when traditional databases have problems.*
Before the digital computer:
- **Natural sciences** rich source of problems, statistical methods developed to solve those problems
- **Very important notion:** well-posed (well-conditioned) problem: solution exists, is unique, and is continuous w.r.t. problem parameters
- *Simply doesn’t make sense to solve ill-posed problems*

Advent of the digital computer:
- *Split in (yet-to-be-formed field of) “Computer Science”*
- Based on application (scientific/numerical computing vs. business/consumer applications) as well as tools (continuous math vs. discrete math)
- *Two very different perspectives on relationship b/w algorithms and data*
Two-step approach for “numerical” problems

- Is problem well-posed/well-conditioned?
- If no, replace it with a well-posed problem. (Regularization!)
- If yes, design a stable algorithm.

View Algorithm A as a function $f$

- Given $x$, it tries to compute $y$ but actually computes $y^*$
- Forward error: $\Delta y = y^* - y$
- Backward error: smallest $\Delta x$ s.t. $f(x + \Delta x) = y^*$
- Forward error $\leq$ Backward error $\times$ condition number
- Backward-stable algorithm provides accurate solution to well-posed problem!
Relationship b/w algorithms and data (3 of 3)

**One-step approach** for study of computation, *per se*

- Concept of computability captured by 3 seemingly-different discrete processes (recursion theory, λ-calculus, Turing machine)
- Computable functions have internal structure (P vs. NP, NP-hardness, etc.)
- Problems of practical interest are “intractable” (e.g., NP-hard vs. poly(n), or \(O(n^3)\) vs. \(O(n \log n)\))

**Modern Theory of Approximation Algorithms**

- provides *forward-error* bounds for *worst-case* input
- worst case in two senses: (1) for all possible input & (2) i.t.o. relatively-simple complexity measures, but independent of “structural parameters”
- get bounds by “relaxations” of IP to LP/SDP/etc., i.e., a “nicer” place
Statistical regularization (1 of 3)

Regularization in statistics, ML, and data analysis

• arose in integral equation theory to "solve" ill-posed problems
• computes a better or more "robust" solution, so better inference
• involves making (explicitly or implicitly) assumptions about data
• provides a trade-off between "solution quality" versus "solution niceness"
• often, heuristic approximation procedures have regularization properties as a "side effect"
• lies at the heart of the disconnect between the "algorithmic perspective" and the "statistical perspective"
Statistical regularization (2 of 3)

Usually implemented in 2 steps:

- add a norm constraint (or “geometric capacity control function”) $g(x)$ to objective function $f(x)$
- solve the modified optimization problem

$$x' = \arg\min_x f(x) + \lambda g(x)$$

Often, this is a “harder” problem, e.g., L1-regularized L2-regression

$$x' = \arg\min_x \|Ax-b\|_2 + \lambda \|x\|_1$$
Regularization is often observed as a side-effect or by-product of other design decisions

- “binning,” “pruning,” etc.
- “truncating” small entries to zero, “early stopping” of iterations
- approximation algorithms and heuristic approximations engineers do to implement algorithms in large-scale systems

BIG question: Can we formalize the notion that/when approximate computation can implicitly lead to “better” or “more regular” solutions than exact computation?
Outline and overview

Preamble: algorithmic & statistical perspectives

General thoughts: data, algorithms, and explicit & implicit regularization

Approximate first nontrivial eigenvector of Laplacian

- Three random-walk-based procedures (heat kernel, PageRank, truncated lazy random walk) are *implicitly solving a regularized optimization exactly*!

Spectral versus flow-based algs for graph partitioning

- Theory says each regularizes in different ways; empirical results agree!

Weakly-local and strongly-local graph partitioning methods

- Operationally like L1-regularization and already used in practice!
Notation for weighted undirected graph

- vertex set $V = \{1, \ldots, n\}$
- edge set $E \subset V \times V$
- edge weight function $w : E \rightarrow \mathbb{R}_+$
- degree function $d : V \rightarrow \mathbb{R}_+$, $d(u) = \sum_v w(u, v)$
- diagonal degree matrix $D \in \mathbb{R}^{V \times V}$, $D(v, v) = d(v)$
- combinatorial Laplacian $L_0 = D - W$
- normalized Laplacian $L = D^{-1/2} L_0 D^{-1/2}$
Approximating the top eigenvector

**Basic idea:** Given an SPSD (e.g., Laplacian) matrix $A$,

- **Power method** starts with $v_0$, and iteratively computes
  \[ v_{t+1} = \frac{Av_t}{||Av_t||_2} \, . \]
- Then, $v_t = \sum_i \gamma_i^+ v_i \rightarrow v_1 \, .$
- If we truncate after (say) 3 or 10 iterations, still have some mixing from other eigen-directions

**What objective does the exact eigenvector optimize?**

- Rayleigh quotient $R(A,x) = x^T A x / x^T x$, for a vector $x$.
- But can also express this as an SDP, for a SPSD matrix $X$.
- *(We will put regularization on this SDP!)*
Views of approximate spectral methods

Three common procedures (L=Laplacian, and M=r.w. matrix):

• **Heat Kernel:**

  \[ H_t = \exp(-tL) = \sum_{k=0}^{\infty} \frac{(-t)^k}{k!} L^k \]

• **PageRank:**

  \[ \pi(\gamma, s) = \gamma s + (1 - \gamma) M \pi(\gamma, s) \]

  \[ R_\gamma = \gamma (I - (1 - \gamma) M)^{-1} \]

• **q-step Lazy Random Walk:**

  \[ W_\alpha^q = (\alpha I + (1 - \alpha) M)^q \]

**Question:** Do these “approximation procedures” exactly optimizing some regularized objective?
Two versions of spectral partitioning

**VP:**

\[
\begin{align*}
\text{min.} & \quad x^T L_G x \\
\text{s.t.} & \quad x^T L_K x = 1 \\
& \quad < x, 1 >_D = 0
\end{align*}
\]

**R-VP:**

\[
\begin{align*}
\text{min.} & \quad x^T L_G x + \lambda f(x) \\
\text{s.t.} & \quad constraints
\end{align*}
\]
Two versions of spectral partitioning

**VP:**

\[
\begin{align*}
\text{min.} & \quad x^T L_G x \\
\text{s.t.} & \quad x^T L_{K_n} x = 1 \\
& \quad \langle x, 1 \rangle_D = 0
\end{align*}
\]

**SDP:**

\[
\begin{align*}
\text{min.} & \quad L_G \circ X \\
\text{s.t.} & \quad L_{K_n} \circ X = 1 \\
& \quad X \succeq 0
\end{align*}
\]

**R-VP:**

\[
\begin{align*}
\text{min.} & \quad x^T L_G x + \lambda f(x) \\
\text{s.t.} & \quad \text{constraints}
\end{align*}
\]

**R-SDP:**

\[
\begin{align*}
\text{min.} & \quad L_G \circ X + \lambda F(X) \\
\text{s.t.} & \quad \text{constraints}
\end{align*}
\]
A simple theorem

\[(F, \eta)\text{-SDP} \quad \min L \cdot X + \frac{1}{\eta} \cdot F(X)\]
\[\text{s.t.} \quad I \cdot X = 1\]
\[X \succeq 0\]

**Theorem**: Let \(G\) be a connected, weighted, undirected graph, with normalized Laplacian \(L\). Then, the following conditions are sufficient for \(X^*\) to be an optimal solution to \((F, \eta)\text{-SDP}\).

- \[X^* = (\nabla F)^{-1} (\eta \cdot (\lambda^* I - L)), \text{ for some } \lambda^* \in \mathbb{R},\]
- \[I \cdot X^* = 1,\]
- \[X^* \succeq 0.\]
Three simple corollaries

\[ F_H(X) = \text{Tr}(X \log X) - \text{Tr}(X) \] (i.e., \text{generalized entropy})

gives scaled \text{Heat Kernel matrix}, with \( t = \eta \)

\[ F_D(X) = -\log \det(X) \] (i.e., \text{Log-determinant})

gives scaled \text{PageRank matrix}, with \( t \sim \eta \)

\[ F_p(X) = (1/p)\|X\|_p^p \] (i.e., \text{matrix p-norm}, for \( p > 1 \))

gives \text{Truncated Lazy Random Walk}, with \( \lambda \sim \eta \)

( \( F(\cdot) \) specifies the algorithm; “number of steps” specifies the \( \eta \) )

Answer: These “approximation procedures” compute regularized versions of the Fiedler vector exactly!
Outline and overview

Preamble: algorithmic & statistical perspectives

General thoughts: data, algorithms, and explicit & implicit regularization

Approximate first nontrivial eigenvector of Laplacian
• Three random-walk-based procedures (heat kernel, PageRank, truncated lazy random walk) are \textit{implicitly} solving a regularized optimization \textit{exactly}!

\textbf{Spectral versus flow-based algs for graph partitioning}
• Theory says \textit{each regularizes in different ways}; empirical results agree!

Weakly-local and strongly-local graph partitioning methods
• Operationally like L1-regularization and already used in practice!
Graph partitioning

A family of combinatorial optimization problems - want to partition a graph’s nodes into two sets s.t.:

• Not much edge weight across the cut (cut quality)
• Both sides contain a lot of nodes

Several standard formulations:

• Graph bisection (minimum cut with 50-50 balance)
• $\beta$-balanced bisection (minimum cut with 70-30 balance)
• $\text{cutsize}/\min\{|A|,|B|\}$, or $\text{cutsize}/(|A||B|)$ (expansion)
• $\text{cutsize}/\min\{|\text{Vol}(A),|\text{Vol}(B)|\}$, or $\text{cutsize}/(|\text{Vol}(A)||\text{Vol}(B)|)$ (conductance or N-Cuts)

All of these formalizations of the bi-criterion are NP-hard!
Networks and networked data

Lots of “networked” data!!

- **technological networks**
  - AS, power-grid, road networks
- **biological networks**
  - food-web, protein networks
- **social networks**
  - collaboration networks, friendships
- **information networks**
  - co-citation, blog cross-postings, advertiser-bidded phrase graphs...
- **language networks**
  - semantic networks...

interaction graph model of networks:
- **Nodes** represent “entities”
- **Edges** represent “interaction” between pairs of entities
### Social and Information Networks

<table>
<thead>
<tr>
<th>Social nets</th>
<th>Nodes</th>
<th>Edges</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPINIONS</td>
<td>75,877</td>
<td>405,739</td>
<td>Who-trusts-whom [35]</td>
</tr>
<tr>
<td>FLICKR</td>
<td>404,733</td>
<td>2,110,078</td>
<td>Photo sharing [21]</td>
</tr>
<tr>
<td>DELICIOUS</td>
<td>147,567</td>
<td>301,921</td>
<td>Collaborative tagging</td>
</tr>
<tr>
<td>CA-DBLP</td>
<td>317,080</td>
<td>1,049,866</td>
<td>Co-authorship (CA) [4]</td>
</tr>
<tr>
<td>CA-cond-mat</td>
<td>21,363</td>
<td>91,286</td>
<td>CA cond-mat [25]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Information networks</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CIT-hep-th</td>
<td>27,400</td>
<td>352,021</td>
<td>hep-th citations [13]</td>
</tr>
<tr>
<td>BLOG-POSTS</td>
<td>437,305</td>
<td>565,072</td>
<td>Blog post links [28]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Web graphs</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>WEB-GOOGLE</td>
<td>855,802</td>
<td>4,291,352</td>
<td>Web graph Google</td>
</tr>
<tr>
<td>WEB-wt10G</td>
<td>1,458,316</td>
<td>6,225,033</td>
<td>TREC WT10G web</td>
</tr>
</tbody>
</table>

| Bipartite affiliation (authors-to-papers) networks |          |             |                                   |
| ATP-DBLP                        | 615,678  | 944,456     | DBLP [25]                         |

| Internet networks              |          |             |                                   |
| AS                             | 6,474    | 12,572      | Autonomous systems                |
| GNUTELLA                       | 62,561   | 147,878     | P2P network [36]                  |

**Table 1:** Some of the network datasets we studied.
Motivation: Sponsored ("paid") Search
Text based ads driven by user specified query

The process:

- Advertisers bids on query phrases.
- Users enter query phrase.
- Auction occurs.
- Ads selected, ranked, displayed.
- When user clicks, advertiser pays!

Also try: barcelona style chair, knoll barcelona chair, More...

- Barcelona Chair: Sale Weekend
  www.PMod.com/Barcelona-Chair - Customer Appreciation Sale! Save 5% on Barcelona Chair + Free S/H.

- Barcelona Chair - Free Shipping
  www.moderncollections.com - Avoid cheap imitations. Our Barcelona Chair offers genuine quality...

- Barcelona Chairs
  BizRate.com - We Offer 2,500+ Chair Choices. Deals On barcelona chairs.

- Classic Barcelona Chair On Sale $599
  funkyssofha.com - All colors available. The Barcelona Chair is a classic piece that...

1 Barcelona Chair - Volo Leather
Ludwig Mies van der Rohe's Barcelona Chair and Stool (1929), originally created to furnish his German Pavilion at the International Exhibition in Barcelona, have come...

2 Barcelona chair - Wikipedia, the free encyclopedia
The Barcelona chair and ottoman was designed by Mies van der Rohe for ... Barcelona Chair; inspired by its predecessors, the campaign and folding chairs ...
Bidding and Spending Graphs

Uses of Bidding and Spending graphs:

- “deep” micro-market identification.
- improved query expansion.

More generally, user segmentation for behavioral targeting.

A “social network” with “term-document” aspects.
Micro-markets in sponsored search

Goal: Find isolated markets/clusters with sufficient money/clicks with sufficient coherence.
Ques: Is this even possible?

- What is the CTR and advertiser ROI of sports gambling keywords?

Diagram:
- 10 million keywords
- 1.4 Million Advertisers
- Categories: Movies Media, Sports, Gambling, Sport videos
What do these networks “look” like?
The “lay of the land”

**Spectral methods** - compute eigenvectors of associated matrices

**Local improvement** - easily get trapped in local minima, but can be used to clean up other cuts

**Multi-resolution** - view (typically space-like graphs) at multiple size scales

**Flow-based methods** - single-commodity or multi-commodity version of max-flow-min-cut ideas

*Comes with strong underlying theory to guide heuristics.*
Comparison of “spectral” versus “flow”

Spectral:
• Compute an eigenvector
• “Quadratic” worst-case bounds
• Worst-case achieved -- on “long stringy” graphs
• Worse-case is “local” property
• Embeds you on a line (or $K_n$)

Flow:
• Compute a LP
• $O(\log n)$ worst-case bounds
• Worst-case achieved -- on expanders
• Worst case is “global” property
• Embeds you in L1

Two methods -- complementary strengths and weaknesses
• What we compute is determined at least as much by as the approximation algorithm as by objective function.
Explicit versus implicit geometry

Explicitly-imposed geometry

• Traditional regularization uses explicit norm constraint to make sure solution vector is “small” and not-too-complex.

Implicitly-imposed geometry

• Approximation algorithms *implicitly* embed the data in a “nice” metric/geometric place and then round the solution.
Regularized and non-regularized communities (1 of 2)

- **Metis+MQI** - a Flow-based method (red) gives sets with better conductance.
- **Local Spectral** (blue) gives tighter and more well-rounded sets.
Regularized and non-regularized communities (2 of 2)

Two ca. 500 node communities from Local Spectral Algorithm:

Two ca. 500 node communities from Metis+MQI:
Outline and overview

Preamble: algorithmic & statistical perspectives

General thoughts: data, algorithms, and explicit & implicit regularization

Approximate first nontrivial eigenvector of Laplacian
  • Three random-walk-based procedures (heat kernel, PageRank, truncated lazy random walk) are *implicitly* solving a regularized optimization *exactly*!

Spectral versus flow-based algs for graph partitioning
  • Theory says each regularizes in different ways; empirical results agree!

Weakly-local and strongly-local graph partitioning methods
  • *Operationally like L1-regularization, and already used in practice!*
Computing locally-biased partitions

Often want clusters “near” a pre-specified set of nodes:

• Large social graphs have good small clusters, don’t have good large clusters
• Might have domain knowledge, so find “semi-supervised” clusters
• As algorithmic primitives, e.g., to solve linear equations fast.
Recall global spectral graph partitioning

The basic optimization problem:

\[
\begin{align*}
\text{minimize} & \quad x^T L_G x \\
\text{s.t.} & \quad \langle x, x \rangle_D = 1 \\
& \quad \langle x, 1 \rangle_D = 0
\end{align*}
\]

- Relaxation of:
  \[
  \phi(G) = \min_{S \subseteq V} \frac{E(S, \bar{S})}{Vol(S) Vol(\bar{S})}
  \]

- Solvable via the eigenvalue problem:
  \[
  \mathcal{L}_G y = \lambda_2(G) y
  \]

- Sweep cut of second eigenvector yields:
  \[
  \lambda_2(G)/2 \leq \phi(G) \leq \sqrt{8\lambda_2(G)}
  \]

Idea to compute locally-biased partitions:

- Modify this objective with a locality constraint
- Show that some/all of these nice properties still hold locally
Local spectral partitioning ansatz

Primal program:

minimize $x^T L_G x$

s.t. $<x, x>_D = 1$

$<x, s>_D^2 \geq \kappa$

Dual program:

max $\alpha - \beta(1 - \kappa)$

s.t. $L_G \geq \alpha L_{K_n} - \beta \left( \frac{L_{K_T}}{\text{vol}(\bar{T})} + \frac{L_{K_{\bar{T}}}}{\text{vol}(T)} \right)$

$\beta \geq 0$

Interpretation:

• Find a cut well-correlated with the seed vector $s$.

• If $s$ is a single node, this relaxes:

$$\min_{S \subseteq V, s \in S, |S| \leq 1/k} \frac{E(S, \bar{S})}{\text{Vol}(S)\text{Vol}(\bar{S})}$$

Interpretation:

• Embedding a combination of scaled complete graph $K_n$ and complete graphs $T$ and $\bar{T}$ ($K_T$ and $K_{\bar{T}}$) - where the latter encourage cuts near $(T, \bar{T})$. 

Mahoney, Orecchia, and Vishnoi (2010)
Main theoretical results

Mahoney, Orecchia, and Vishnoi (2010)

**Theorem:** If $x^*$ is an optimal solution to LocalSpectral,

(*) it is a Generalized Personalized PageRank vector, and can be computed as solution to a set of linear equations;

(*) one can find a cut of conductance $\leq 8\lambda(G,s,\kappa)$ in time $O(n \lg n)$ with sweep cut of $x^*$;

(*) For all sets of nodes $T$ s.t. $\kappa' := \langle s, s_T \rangle_D^2$, we have: $\phi(T) \geq \lambda(G,s,\kappa)$ if $\kappa \leq \kappa'$, and $\phi(T) \geq (\kappa'/\kappa)\lambda(G,s,\kappa)$ if $\kappa' \leq \kappa$.

Fast running time guarantee.

Upper bound, as usual from sweep cut & Cheeger.

Lower bound: Spectral version of flow-improvement algs.
Illustration on small graphs

Mahoney, Orecchia, and Vishnoi (2010)

• Similar results if we do local random walks, truncated PageRank, and heat kernel diffusions.

• Often, it finds “worse” quality but “nicer” partitions than flow-improve methods. (Tradeoff we’ll see later.)
A somewhat different approach

**Strongly-local spectral methods**

- ST04: truncated "local" random walks to compute locally-biased cut
- ACL06: approximate locally-biased PageRank vector computations
- Chung08: approximate heat-kernel computation to get a vector

These are the diffusion-based procedures that we saw before

*except truncate/round/clip/push small things to zero* starting with localized initial condition

Also get provably-good local version of global spectral
What's the connection?

“Optimization” approach:
- Well-defined objective $f$
- Weakly local (touch all nodes), so good for medium-scale problems
- Easy to use

“Operational” approach*:
- Very fast algorithm
- Strongly local (clip/truncate small entries to zero), good for large-scale
- Very difficult to use

* Informally, optimize $f+\lambda g$ (... almost formally!): steps are structurally-similar to the steps of how, e.g., L1-regularized L2 regression algorithms, implement regularization

More importantly,
- This “operational” approach is already being adopted in PODS/VLDB/SIGMOD/KDD/WWW environments!
- Let’s make the regularization explicit—and know what we compute!
Looking forward ...

A common *modus operandi* in many (really*) large-scale applications is:

- Run a procedure that bears some resemblance to the procedure you would run if you were to solve a given problem exactly
- Use the output in a way similar to how you would use the exact solution, or prove some result that is similar to what you could prove about the exact solution.

**BIG Question:** Can we make this more principled? E.g., can we “engineer” the approximations to solve (exactly but implicitly) some regularized version of the original problem---to do large scale analytics in a statistically more principled way?

*e.g., industrial production, publication venues like WWW, SIGMOD, VLDB, etc.*
Conclusions

Regularization is:

• absent from CS, which historically has studied computation per se
• central to nearly area that applies algorithms to noisy data
• gets at the heart of the algorithmic-statistical “disconnect"

Approximate computation, in and of itself, can implicitly regularize:

• Theory & the empirical signatures in matrix and graph problems
• Solutions of approximation algorithms don’t need to be something we “settle for,” they can be “better” than the “exact” solution

In very large-scale analytics applications:

• Can we “engineer” database operations so “worst-case” approximation algorithms exactly solve regularized versions of original problem?
• I.e., can we get best of both worlds for very large-scale analytics?
**Objectives:**

- Address algorithmic, statistical, and mathematical challenges in modern statistical data analysis.

- Explore novel techniques for modeling and analyzing massive, high-dimensional, and nonlinearly-structured data.

- Bring together computer scientists, statisticians, mathematicians, and data analysis practitioners to promote cross-fertilization of ideas.

**Organizers:** M. W. Mahoney, A. Shkolnik, G. Carlsson, and P. Drineas,

*Registration is available now!*