Laplacian Paradigm 2.0

8:40-9:10: Merging Continuous and Discrete (Richard Peng)
9:10-9:50: Beyond Laplacian Solvers (Aaron Sidford)
9:50-10:30: Approximate Gaussian Elimination (Sushant Sachdeva)
10:30-11:00: coffee break
11:00-12:00: Analysis using matrix Martingales (Rasmus Kyng)
12:00-14:00 lunch
14:00-15:00 Graph Structure via Eliminations (Aaron Schild)

Website: bit.ly/laplacian2
Merging the Continuous and Discrete

Richard Peng

Oct 6, 2018
Outline

- Graphs and Laplacians
- Building Blocks
- Laplacian Paradigm 2.0
Large Networks

- Data mining: centrality, clustering...
- Image/video processing: segmentation, denoising ...
- Scientific computing: stress, fluids, waves...
- \( \backslash \) (linear system solve)
- CVX (convex optimization)
- Eigenvector solvers
Graphs and Matrices

High performance computing: non-zeros $\Leftrightarrow$ edges, 
design / analyze matrix algorithms using graph theory

$$
\begin{bmatrix}
2 & -1 & -1 \\
-1 & 1 & 0 \\
-1 & 0 & 1 \\
\end{bmatrix}
$$

n vertices
m edges

n rows / columns
O(m) non-zeros
Graphs and Matrices

High performance computing: non-zeros $\leftrightarrow$ edges, design / analyze matrix algorithms using graph theory

$n$ rows / columns
$O(m)$ non-zeros

Graph Laplacian matrix $L$
- Diagonal: degrees
- Off-diagonal: -edge weights

$n$ vertices
$m$ edges
Source of Laplacians

\[
\begin{pmatrix}
2 & -1 & -1 \\
-1 & 1 & 0 \\
-1 & 0 & 1
\end{pmatrix}
\]

graph Laplacian matrix \( L \)

- Diagonal: degrees
- Off-diagonal: -edge weights

d-Regular graphs: \( L = dI - A \), \( A \): adjacency matrix
Source of Laplacians

Graph cuts: $\mathbf{x}^T \mathbf{L} \mathbf{x} = \sum_{u \sim v} \mathbf{w}_{uv} (\mathbf{x}_u - \mathbf{x}_v)^2$

- $x_a = 1$
- $x_b = 0$
- $x_c = 1$

$(1-0)^2 = 1$
$(1-1)^2 = 0$

$\mathbf{L}$ is the graph Laplacian matrix, with:
- Diagonal: degrees
- Off-diagonal: edge weights

$d$-Regular graphs: $\mathbf{L} = d\mathbf{I} - \mathbf{A}$, $\mathbf{A}$: adjacency matrix

$x$ indicator vector of cut $\Rightarrow$ weight of cut
Source of Laplacians

\[ L = B^TWB \] where \( B \) is edge-vertex incidence matrix

Graph cuts: \( x^T L x = \sum_{u \sim v} w_{uv} (x_u - x_v)^2 \)

- \( x_a = 1 \)
- \( x_b = 0 \)
- \( x_c = 1 \)

\[ (1-0)^2 = 1 \]
\[ (1-1)^2 = 0 \]

- \( B \) is edge-vertex incidence matrix
- \( \sum_{u \sim v} \) weight of cut

Graph Laplacian matrix \( L \)
- Diagonal: degrees
- Off-diagonal: -edge weights
Origin of the Laplacian Paradigm

[Spielman Teng `04]
Input: graph Laplacian $\mathbf{L}$
  vector $\mathbf{b}$
Output: vector $\mathbf{x}$ s.t. $\mathbf{Lx} \approx \mathbf{b}$
Runtime: $O(m\log^{O(1)} n\log(1/\epsilon))$
Origin of the Laplacian Paradigm

[Spielman Teng `04]
Input: graph Laplacian \( \mathbf{L} \)
vector \( \mathbf{b} \)
Output: vector \( \mathbf{x} \) s.t. \( \mathbf{Lx} \cong \mathbf{b} \)
Runtime: \( O(m\log \Omega(1) + \log(1/\epsilon)) \)

[Cohen-Kyang-Miller-Pachocki-P-Rao-Xu `14]: \( \leq 1/2 \)
Spielman Teng `04
Input: graph Laplacian $L$
vector $b$
Output: vector $x$ s.t. $Lx \approx b$
Runtime: $O(m \log n \log(1/\varepsilon))$

[Cohen-Kyng-Miller-Pachocki-P-Rao-Xu `14]: $\leq 1/2$

Origin of the Laplacian Paradigm
The Laplacian Paradigm

Directly related:
Elliptic systems

Few iterations:
Eigenvalues, Heat kernels

Many iterations / modify algorithm
Graph problems Image processing
Outline

- Graphs and Laplacians
- **Building Blocks**
- Laplacian Paradigm 2.0
**Lx = b** as a graph problem

**x:** voltage vectors  
**Dual:** electrical flow **f**

Unified formulation: \( \min_{\|f\|_p} \) with residual \( b \) \( \|f\|_p \):

- \( p = 2 \): solving \( Lx = b \)
- \( p = 1 \): shortest path / transshipment
- \( p = \infty \): max-flow/min-cut
Direct Methods (combinatorial)

Repeatedly remove vertices by creating equivalent graphs on their neighborhoods

\[ M^{(2)} \leftarrow \text{Eliminate}(M^{(1)}, i_1) \]
\[ M^{(3)} \leftarrow \text{Eliminate}(M^{(2)}, i_2) \]

...
Direct Methods (combinatorial)

Repeatedly remove vertices by creating equivalent graphs on their neighborhoods

\[
\mathbf{M}^{(2)} \leftarrow \text{Eliminate}(\mathbf{M}^{(1)}, i_1)
\]
\[
\mathbf{M}^{(3)} \leftarrow \text{Eliminate}(\mathbf{M}^{(2)}, i_2)
\]
...

- Parallel graph algorithms
- Matrix multiplication / dense solves
- Sparsified squaring
Iterative Methods (numerical)

Solve $Ax = b$ by

$x \leftarrow x - (Ax - b)$

Fixed point: $Ax - b = 0$
Iterative Methods (numerical)

Preconditioning:
Solve $B^{-1}Ax = B^{-1}b$ by:

$x \leftarrow x - B^{-1}(Ax - b)$

Fixed point: $Ax - b = 0$

- Simple $B$: $B = I$, many iterations
- $B = A$: 1 iteration, but same problem
Iterative Methods (numerical)

Preconditioning:
Solve $B^{-1}Ax = B^{-1}b$ by:

$$x \leftarrow x - B^{-1}(Ax - b)$$

Fixed point: $Ax - b = 0$

- Simple $B$: $B = I$, many iterations
- $B = A$: 1 iteration, but same problem

- Krylov space methods / PCG
- Convex optimization algorithms
Hard instances

Direct methods create too much fill on highly connected graphs

Iterative methods take too many iterations paths
Hard instances

Direct methods create too much fill on highly connected graphs

Iterative methods take too many iterations paths

Still ‘easy’ by themselves

Easy for iterative methods

Easy for direct methods
Hard instances

Direct methods create too much fill on highly connected graphs

Iterative methods take too many iterations paths

Still ‘easy’ by themselves

Easy for iterative methods

Easy for direct methods

Must handle both simultaneously, but avoid paying \( n \) iterations \( \times m \) per iteration
Hybrid algorithms (aka. v1.0)

- Scientific computing: iChol, multigrid
- [Vaidya `89] precondition with graphs
Hybrid algorithms (aka. v1.0)
• Scientific computing: iChol, multigrid
• [Vaidya `89] precondition with graphs
Hybrid algorithms (aka. v1.0)
• Scientific computing: iChol, multigrid
• [Vaidya ‘89] precondition with graphs

Focus: how to combine
• [Gemban-Miller ‘96]: spectral graph theory
• [Spielman-Teng ‘04]: spectral (ultra-)sparsify
Key “glue”: sparsification

[Spielman-Teng `04]: for any G, can find H with \( O(n \log^{O(1)} n) \) edges s.t. \( x^T L_G x \approx x^T L_H x \ \forall x \)
Key “glue”: sparsification

[Spielman-Teng `04]: for any G, can find H with $O(\log^{O(1)} n)$ edges s.t. $x^T L_G x \approx x^T L_H x \; \forall x$

- Combinatorial parameter: #edges
- Numerical parameter : approximations
Key “glue”: sparsification

[Spiegelman-Teng `04]: for any G, can find H with $O(n \log^{O(1)} n)$ edges s.t. $x^T L_G x \approx x^T L_H x \ \forall x$

- Combinatorial parameter: #edges
- Numerical parameter: approximations

[Spiegelman-Srivatava `08]: sample by effective resistances gives H with $O(n \log n)$ edges
Outline

- Graphs and Laplacians
- Building Blocks
- Laplacian Paradigm 2.0
Max-Flow problem

Maximum number of disjoint s-t paths

Applications:
• Routing
• Scheduling

Recall: \( \min_{f \text{ with residual } b} \|f\|_p \):
• \( p = 2 \): solving \( Lx = b \)
• \( p = \infty \): max-flow/min-cut
Max-Flow problem

Maximum number of disjoint s-t paths

Dual: separate s and t by removing fewest edges

Applications:
- Routing
- Scheduling

Recall: \( \min_f \text{ with residual } b \|f\|_p: \)
- \( p = 2: \) solving \( Lx = b \)
- \( p = \infty: \) max-flow/min-cut

Applications:
- Partitioning
- Clustering
Hybrid Algorithms for Max-Flow

[Daitch-Spielman `08][Christiano-Kelner-Madry-Spielman-Teng `10]:
[Lee-Sidford `14] Max-flow/Min-cut via (several) electrical flows

Repeat about $m^{1/3}$ iters
• Solve linear systems
• Re-adjust edge weights

$Lx = b$
Hybrid Algorithms for Max-Flow

[Daitch-Spielman `08][Christiano-Kelner-Madry-Spielman-Teng `10]:
[Lee-Sidford `14] Max-flow/Min-cut via (several) electrical flows

Repeat about $m^{1/3}$ iters
- Solve linear systems
- Re-adjust edge weights

[Madry `10] [Racke-Shah-Taubig `14]:
cut approximator / oblivious routing
$O(n^{o(1)})$-approx. in $O(m^{1+o(1)})$
Hybrid Algorithms for Max-Flow

[Daitch-Spielman `08][Christiano-Kelner-Madry-Spielman-Teng `10]:

[Lee-Sidford `14] Max-flow/Min-cut via (several) electrical flows

Repeat about $m^{1/3}$ iters
• Solve linear systems
• Re-adjust edge weights

[Lee-Rao-Srivastava `13][Sherman `13, `17][Kelner-Lee-Orecchia-Sidford `14]:
Preconditioning, $(1+\varepsilon)$-approx

[Madry `10] [Racke-Shah-Taubig `14]:
cut approximator / oblivious routing
$O(n^{o(1)})$-approx. in $O(m^{1+o(1)})$
Hybrid Algorithms for Max-Flow

[Daitch-Spielman `08][Christiano-Kelner-Madry-Spielman-Teng `10]:

[Lee-Sidford `14] Max-flow/Min-cut via (several) electrical flows

Repeat about $m^{1/3}$ iters
- Solve linear systems
- Re-adjust edge weights

[Lee-Rao-Srivastava `13][Sherman `13, `17][Kelner-Lee-Orecchia-Sidford `14]:
Preconditioning, $(1+\epsilon)$-approx

[P`16]: recurse them into each other:
$O(m\log^{41}n)$, optimistically $m\log^{6}n$

[Madry `10] [Racke-Shah-Taubig `14]:
cut approximator / oblivious routing
$O(n^{o(1)})$-approx. in $O(m^{1+o(1)})$
Laplacian Paradigm 2.0

Motivated by the goal of hybrid algorithms, modify direct and iterative methods

New Intermediate structures / theorems motivated by the overall algorithms
Examples

Directed graphs / asymmetric matrices

Sparsified/Approximate Gaussian Elimination

SC(A, C)
Under the hood

Matrix (martingale) concentration

Partitioning / Localizations of Random Walks

\[ \text{G}[V_1] \quad + \quad \text{Sc}(G, V_2) \]
Not covered 😞

Matrix Zoo from Scientific Computing

[Boman-Hendrickson-Vavasis `04]
[Kyng-Lee-P-Sachdeva-Spielman `16]
[Kyng-Zhang `17][Kyng-P-Schweiterman-Zhang `18]

Interactions with data structures

[Kelner-Orecchia-Sidford-Zhu `13]
[Nanongkai-Saranuk `17][Wulff-Nilsen `17]
[Durfee-Kyng-Peebles-Rao-Sachdeva `17]
Questions

<table>
<thead>
<tr>
<th>Wist list</th>
<th>Direct</th>
<th>Iterative</th>
<th>Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex functions</td>
<td>☹</td>
<td>☑️</td>
<td>☑️??</td>
</tr>
<tr>
<td>Arbitrary values</td>
<td>☑️</td>
<td>☹</td>
<td>☹️??</td>
</tr>
<tr>
<td>Dynamic/streaming</td>
<td>☑️</td>
<td>☹️</td>
<td>☑️???</td>
</tr>
</tbody>
</table>

Approximate eliminations beyond spectral condition #

Unreasonable effectiveness of pcg(ichol(A), b), multigrid

Non-linear (preconditioned) iterative methods

[Adil-Kyng-P-Sachdeva `19]:
p-norm iterative refinement
Solvers in Practice

[Kyng-Rao-Sachdeva `15] we suggest rerunning the program a few times... An alternate solver based on iChol is provided...

Questions:
- Precision
- (pseudo) deterministic