Laplacian Gems

Daniel A. Spielman
Yale University
Outline

Laplacian Linear Systems (Electrical Graph Theory)

Applications

Fast Algorithms
  Bold ideas
  Surprising results
  Powerful techniques

Faster than Koutis-Miller-Peng?
Graphs as Spring Networks

edges  -> ideal linear springs
weights  -> spring constants (k)

Physics: when stretched to length x, force is kx
potential energy is kx^2/2

Nail down some vertices, let rest settle
Graphs as Spring Networks

Nail down some vertices, let rest settle

Physics: minimizes total potential energy

\[
\sum_{(i,j) \in E} w_{i,j} (v_i - v_j)^2 = v^T L v
\]

subject to boundary constraints (nails)
Tutte’s Theorem ‘63

If nail down a face of a planar 3-connected graph, get a planar embedding!
The Laplacian

\[ v^T L v = \sum_{(i,j) \in E} w_{i,j} (v_i - v_j)^2 \]

\[ L_{i,j} = \begin{cases} 
-w_{i,j} & \text{if } (i, j) \in E \\
 0 & \text{otherwise} \\
 d_i & \text{if } i = j 
\end{cases} \]

\[ d_i = \sum_j w_{i,j} \]

\[ L = \sum_{(i,j) \in E} w_{i,j} (e_i - e_j)(e_i - e_j)^T \]
The Laplacian

\[ v^T L v = \sum_{(i,j) \in E} w_{i,j} (v_i - v_j)^2 \]

\[ L_{i,j} = \begin{cases} 
-w_{i,j} & \text{if } (i, j) \in E \\
d_i & \text{if } i = j \\
0 & \text{otherwise} 
\end{cases} \]

Can minimize, subject to boundary \( v_i = f_i \) for \( i \in S \) by Laplacian Linear Solvers in nearly-linear time

[Koutis-Miller-Peng ‘10, S-Teng ‘04]
Regression on Graphs [Zhu-Ghahramani-Lafferty ’03]

Say know $f(i)$ for all $i$ in $S \subseteq V$ and want to guess $f(j)$ for all other $j$.

Idea: nail node $i$ to $f(i)$ on Real line.

$$\min \sum_{(i,j) \in E} w_{i,j} (v_i - v_j)^2 = v^T L v$$
Graphs as Resistor Networks

edge  -> resistor
weight -> 1/resistance

Ohm’s law: flow/current

\[ f_{i,j} = \frac{v_i - v_j}{r_{i,j}} = w_{i,j}(v_i - v_j) \]
Graphs as Resistor Networks

edge   -> resistor
weight -> 1/resistance

Ohm’s law: flow/current

\[ f_{i,j} = \frac{v_i - v_j}{r_{i,j}} = w_{i,j}(v_i - v_j) \]

Demands/external current:

\[ d = Lv \quad v = L^+ d \]

*the pseudo-inverse*
Graphs as Resistor Networks

Ohm’s law: flow/current

\[ f_{i,j} = \frac{v_i - v_j}{r_{i,j}} = w_{i,j}(v_i - v_j) \]

Demands/external current:

\[ d = Lv \quad v = L^+d \]

To flow 1 from s to t:

Set \( d(s) = 1 \) and \( d(t) = -1 \)

Solve for \( v \)

Use Ohm’s law to find flow in graph
Graphs as Resistor Networks

To solve for current when fix voltages

Solve for $v$ minimizing dissipated energy,

$$v^T L v = \sum w_{i,j} (v_i - v_j)^2$$

subject to fixed potentials
Graphs as Resistor Networks

To solve for current when fix voltages

Solve for $v$ minimizing dissipated energy,

$$v^T Lv = \sum w_{i,j} (v_i - v_j)^2$$

subject to fixed potentials
Effective Resistance between s and t

= resistance of whole network between s and t

= $v_s - v_t$ in s-t flow of value 1

= $(e_s - e_t)^T L^+ (e_s - e_t)$

$$f_{i,j} = \frac{v_i - v_j}{r_{i,j}}$$
Effective Resistance between s and t

= resistance of whole network between s and t

= $v_s - v_t$ in s-t flow of value 1

= $(e_s - e_t)^T L^+ (e_s - e_t)$

= $1$/effective spring constant between s and t

= prob edge (s,t) in a random spanning tree

~ expected commute time between s and t

[Chandra-Raghavan-Ruzzo-Smolensky-Tiwari ‘89]

And, is a distance.
Application: Random Spanning Trees

Can use fast Laplacian solvers to sample random spanning trees in time $\tilde{O}(mn^{1/2})$ [Kelner-Madry-Propp ‘09]

Idea: accelerate random-walk based sampling [Aldous, Broder]

by jumping through clusters where walk is slow.
Application: Approximate Maximum Flow

[Christiano-Kelner-Madry-S-Teng ’10]

1. Compute electrical s-t flow
2. Increase resistance on over-capacity edges
3. Return average of computed flows (a la MWM)

Time $\tilde{O}(m^{3/2}/\epsilon^3)$
Application: Approximate Maximum Flow

[Christiano-Kelner-Madry-S-Teng ’10]

1. Compute electrical s-t flow

2. Increase resistance on over-capacity edges and delete very over-capacity edges

3. Return average of computed flows (a la MWM)

Time: \( \tilde{O}(m^{3/2}/\epsilon^3) \rightarrow \tilde{O}(mn^{1/3}/\epsilon^{11/3}) \)
Approximate Laplacian Solvers

<table>
<thead>
<tr>
<th>Method</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preconditioned Conjugate Gradient</td>
<td>$O(mn)$</td>
</tr>
<tr>
<td>[Hestenes ‘51, Stiefel ‘52, ???]</td>
<td></td>
</tr>
<tr>
<td>Vaidya ‘90: Subgraph preconditioners</td>
<td>$O(mn^{3/4})$</td>
</tr>
<tr>
<td>Boman-Hendrickson ‘01:</td>
<td>$\tilde{O}(mn^{1/2})$</td>
</tr>
<tr>
<td>Using Low-Stretch Spanning Trees</td>
<td></td>
</tr>
<tr>
<td>S-Teng ‘04: Spectral sparsification</td>
<td>$\tilde{O}(m)$</td>
</tr>
<tr>
<td>Koutis-Miller-Peng ‘10: Elegance</td>
<td>$O(m \log^2 n)$</td>
</tr>
</tbody>
</table>
Preconditioned Conjugate Gradient

Solve $L_G x = b$ by

Approximating $L_G$ by $L_H$ (the preconditioner)

In each iteration

solve a system in $L_H$

multiply a vector by $L_G$

$\epsilon$-approx solution after

$$O\left(\sqrt{\kappa(L_G, L_H)} \log \epsilon^{-1}\right)$$ iterations

condition number/approx quality
Graph Sparsification [Benczur-Karger ‘96]

Approximate G by a sparse H, approximately preserving all cuts

$(1 \pm \epsilon)$
Spectral Sparsification [S-Teng ‘04]

Approximate $G$ by a sparse $H$, approximately preserving all energies and effective resistances

$$v^T L_G v \approx v^T L_H v (1 \pm \epsilon)$$
Spectral Sparsification [S-Srivastava ‘08]

\[ u^T L_G u \rightarrow u^T L_H u \]

\[ (1 \pm \epsilon) \]

Sample edges \((i,j)\) of \(G\) with probability proportional to \(R_{eff}(i,j)w_{i,j}\)

\[ O(n \log n/\epsilon^2) \] edges suffice

Analysis by random matrix concentration results of Rudelson and Vershynin.
Spectral Sparsification [Koutis-Miller-Peng ‘10]

Understand Rudelson-Vershynin much better

Suffices to sample edges \((i,j)\) of \(G\) with probability crudely related to \(R_{eff}(i,j)w_{i,j}\)

Sample \(O(n \log n/\epsilon^2)\) edges \textit{with replacement} so actually get many fewer edges!
Local Graph Clustering [S-Teng ‘04]

Given vertex of interest
find nearby cluster $S$
with small conductance
in time $O(|S|)$
Local Graph Clustering [S-Teng ‘04]

Given vertex of interest
find nearby cluster $S$
with small conductance
in time $O(|S|)$

How should one explore a graph?
How should one explore a graph?

Possible goals:

- find nodes like the first node
- find a representative sample
- minimize computation and graph access
- prove something rigorous
How should one explore a graph?

Possible goals:

find nodes like the first node

find a representative sample

minimize computation and graph access

prove something rigorous

Beat BFS
Local Graph Clustering [S-Teng ‘04]

Prove: if $S$ has small conductance $\phi$

- $u$ is a random node in $S$
- probably
  - find a set of small conductance, $\phi^{1/2} \log^c n$
  - in time $|S| \log^c n/\phi^2$
Using Approximate Personal PageRank Vectors

Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

Spilling paint in a graph:
start at one node
at each step,
\( \alpha \) fraction dries
of wet paint, half stays put, half to neighbors
Using Approximate Personal PageRank Vectors

Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

Spilling paint in a graph:
start at one node
at each step,
\( \alpha \) fraction dries
of wet paint, half stays put, half to neighbors
Using Approximate Personal PageRank Vectors

Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

Spilling paint in a graph:
start at one node
at each step,
\(\alpha\) fraction dries
of wet paint, half stays put, half to neighbors

with \(\alpha = 1/2\)
Using Approximate Personal PageRank Vectors

Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

Spilling paint in a graph:
start at one node
at each step, $\alpha$ fraction dries
of wet paint, half stays put, half to neighbors

with $\alpha = 1/2$
Using Approximate Personal PageRank Vectors

Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

Spilling paint in a graph:
start at one node
at each step, \( \alpha \) fraction dries
of wet paint, half stays put, half to neighbors

with \( \alpha = 1/2 \)
Using Approximate Personal PageRank Vectors

Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

Spilling paint in a graph:
start at one node
at each step, \( \alpha \) fraction dries

of wet paint, half stays put, half to neighbors

with \( \alpha = 1/2 \)
Using Approximate Personal PageRank Vectors

Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

Spilling paint in a graph:
start at one node
at each step,
\( \alpha \) fraction dries
of wet paint, half stays put, half to neighbors

Time doesn’t matter, can push paint whenever

Approximate: only push when a lot of paint
Using Approximate Personal PageRank Vectors

Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

Spilling paint in a graph:
- start at one node
- at each step, \( \alpha \) fraction dries
- of wet paint, half stays put, half to neighbors

Time doesn’t matter, can push paint whenever

Approximate: only push when a lot of paint
Using Approximate Personal PageRank Vectors

Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

Spilling paint in a graph:
start at one node
at each step,
α fraction dries
of wet paint, half stays put, half to neighbors

Time doesn’t matter, can push paint whenever

Approximate: only push when a lot of paint
Using Approximate Personal PageRank Vectors

Jeh-Widom ‘03, Berkhin ‘06, Andersen-Chung-Lang ’06

Spilling paint in a graph:
start at one node
at each step,
α fraction dries
of wet paint, half stays put, half to neighbors

Time doesn’t matter, can push paint whenever

Approximate: only push when a lot of paint
Spilling paint in a graph:
start at one node
at each step,
a fraction dries
of wet paint, half stays put, half to neighbors

Time doesn’t matter, can push paint whenever

Approximate: only push when a lot of paint
Volume-Biased Evolving Set Markov Chain

[Andersen-Peres ’09]

Walk on sets of vertices
starts at one vertex, ends at V

Dual to random walk on graph

When start inside set of conductance $\phi$
find set of conductance $\phi^{1/2} \log^{1/2} n$
with work $|S| \log^c n/\phi^{1/2}$
Volume-Biased Evolving Set Markov Chain

[Andersen-Peres ‘09]

Walk on sets of vertices
starts at one vertex, ends at V

Dual to random walk on graph

When start inside set of conductance $\phi$
find set of conductance $\phi^{1/2} \log^{1/2} n$

with work $|S| \log^c n/\phi^{1/2}$

\textit{can we eliminate this?}
A Technique for proving convergence of walks on graphs of Lovász and Simonovits

Consider lazy random walk on a regular graph (stays put with probability $\frac{1}{2}$).

Approaches uniform in time $1/$conductance$^2$. 
A Technique for proving convergence of walks on graphs of Lovász and Simonovits

Consider lazy random walk on a regular graph (stays put with probability $\frac{1}{2}$).

Approaches uniform in time $1/\text{conductance}^2$.

$$p^t(i) = \text{prob of vertex i at time t}$$

$$C^t(k) = \max_{|S|=k} \sum_{i \in S} p^t(i)$$

sum of largest $k$ values
Lovász-Simonovits

\[ C^t(k) = \max_{|S|=k} \sum_{i \in S} p^t(i) \]
Lovász-Simonovits

\[ C^t(k) = \max_{|S|=k} \sum_{i \in S} p^t(i) \]

extend by linearity
Lovász-Simonovits

\[ C^t(k) = \max_{|S|=k} \sum_{i \in S} p^t(i) \]

interpolate linearly

plot for uniform dist
Lovász-Simonovits

\[ C^t(k) = \max_{|S|=k} \sum_{i \in S} p^t(i) \]

is concave
Lovász-Simonovits: Easy inequality

\[ C^{t+1} \leq C^t \]
Lovász-Simonovits: Conductance $\phi$ inequality

$$C^{t+1}(k) \leq \frac{1}{2} \left( C^t(k - \phi k) + C^t(k + \phi k) \right) \quad k \in [0, n/2]$$

$$C^{t+1}(k) \leq \frac{1}{2} \left( C^t(k - \phi(n - k)) + C^t(k + \phi(n - k)) \right) \quad k \in [n/2, n]$$
Lovász-Simonovits: Conductance $\phi$ inequality

\[ C^{t+1}(k) \leq \frac{1}{2} \left(C^t(k - \phi k) + C^t(k + \phi k)\right) \quad k \in [0, n/2] \]

\[ C^{t+1}(k) \leq \frac{1}{2} \left(C^t(k - \phi(n - k)) + C^t(k + \phi(n - k))\right) \quad k \in [n/2, n] \]
Lovász-Simonovits: Conductance $\phi$ inequality

$$C^{t+1}(k) \leq \frac{1}{2} \left( C^t(k - \phi k) + C^t(k + \phi k) \right) \quad k \in [0, n/2]$$

$$C^{t+1}(k) \leq \frac{1}{2} \left( C^t(k - \phi(n - k)) + C^t(k + \phi(n - k)) \right) \quad k \in [n/2, n]$$
Lovász-Simonovits: Conductance $\phi$ inequality

$$C^{t+1}(k) \leq \frac{1}{2} \left( C^t(k - \phi k) + C^t(k + \phi k) \right) \quad k \in [0, n/2]$$

$$C^{t+1}(k) \leq \frac{1}{2} \left( C^t(k - \phi(n - k)) + C^t(k + \phi(n - k)) \right) \quad k \in [n/2, n]$$
Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West ‘91]
Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West ‘91]
Low-Stretch Spanning Trees (unweighted case)

[Alon-Karp-Peleg-West ‘91]

\[ \text{stretch}_T(i, j) = \text{dist}_T(i, j) \]
Low-Stretch Spanning Trees (unweighted case)

[Alon-Karp-Peleg-West ‘91]

\[
\text{stretch}_T(G) = \sum_{(i,j) \in G} \text{dist}_T(i,j)
\]
Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West ‘91]

\[ \text{stretch}_T(G) = \sum_{(i,j) \in G} \frac{\text{dist}_T(i,j)}{\text{length}(i,j)} \]

Every graph has a spanning tree of low stretch
Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West ‘91]

\[ \text{stretch}_T(G) = \sum_{(i,j) \in G} \frac{\text{dist}_T(i,j)}{\text{length}(i,j)} \]

Every graph has a spanning tree of low stretch

Expander:
Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West ‘91]

\[\text{stretch}_T(G) = \sum_{(i,j) \in G} \frac{\text{dist}_T(i,j)}{\text{length}(i,j)}\]

Every graph has a spanning tree of low stretch

Expander:
- low diameter
- use shortest path tree
Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West ‘91]

\[
stretch_T(G) = \sum_{(i,j) \in G} \frac{\text{dist}_T(i,j)}{\text{length}(i,j)}
\]

Every graph has a spanning tree of low stretch

\[\leq m2^O(\sqrt{\log n \log \log n})\]
Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West ‘91]

Every graph has a spanning tree of low stretch

\[ \leq m^{2^{O(\sqrt{\log n \log \log n})}} \]

Proof: by BFS/shortest paths.
Grow balls until low boundary (Awerbuch)
Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West ‘91]

Proof: by BFS/shortest paths.
Grow balls until low boundary (Awerbuch)
Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West ‘91]

Proof: by BFS/shortest paths.
Grow balls until low boundary (Awerbuch)

Use shortest path trees inside balls.
Low-Stretch Spanning Trees

[Alon-Karp-Peleg-West ‘91]

Proof: by BFS/shortest paths.
Grow balls until low boundary (Awerbuch)

Use shortest path trees inside balls.
Contract balls.
Lower-Stretch Spanning Trees
[Elkin-Emek-S-Teng ‘04, Abraham-Bartal-Neiman ‘08]

Proof: BFS cones centered on a BFS ball.

\[ \leq O(m \log n \log \log n (\log \log \log n)^2) \]
Lower-Stretch Spanning Trees
[Elkin-Emek-S-Teng ‘04, Abraham-Bartal-Neiman ‘08]

Proof: BFS cones centered on a BFS ball.

\[ \leq O(m \log n \log \log n (\log \log \log n)^2) \]

Can we get \( m \log_2 n \)?
Algebraic computation of stretch [S-Woo ’09]

$$\text{stretch}_T(G) = \text{Trace} \left[ L_T^+ L_G \right]$$

Key: in trees, resistance acts like length.

$$R_{\text{eff}}(i, j) = r_1 + r_2$$
Algebraic computation of stretch \([S-Woo \ '09]\)

\[
\text{stretch}_T(G) = \text{Trace} \left[ L_T^+ L_G \right]
\]

\[
\text{Trace} \left[ L_T^+ L_G \right] = \text{Trace} \left[ L_T^+ \sum w_{i,j} (e_i - e_j)(e_i - e_j)^T \right]
\]

\[
= \sum w_{i,j} \text{Trace} \left[ L_T^+ (e_i - e_j)(e_i - e_j)^T \right]
\]

\[
= \sum w_{i,j} \text{Trace} \left[ (e_i - e_j)^T L_T^+ (e_i - e_j) \right]
\]

\[
= \sum w_{i,j} (e_i - e_j)^T L_T^+ (e_i - e_j)
\]

\[
= \sum w_{i,j} R_T^{eff} (i, j)
\]

\[
= \sum w_{i,j} \text{dist}_T (i, j)
\]

\[
= \sum \text{dist}_T (i, j) / \text{length}(i, j)
\]
Preconditioning: solve in time $O(mk)$

Need $H \subseteq G$, $H$ is a tree + $m/k$ edges such that

$$\lambda_{max}(L_H^+ L_G) \leq k^2$$
Preconditioning: solve in time $O(m \, k)$

Need $H \subseteq G$, $H$ is a tree + $m/k$ edges such that $\lambda_{max}(L_H^+ L_G) \leq k^2$

For low-stretch $T$,

$$\sum \lambda_i(L_T^+ L_G) = \text{Trace} \left[ L_T^+ L_G \right] \leq O(m \log n)$$

At most $s$ eigenvalues larger than $O((m/s) \log n)$
Preconditioning: solve in time $O(mk)$

Need $H \subseteq G$, $H$ is a tree + $m/k$ edges such that $\lambda_{max}(L_H^+L_G) \leq k^2$

For low-stretch $T$,
At most $s$ eigenvalues larger than $O((m/s) \log n)$

Kolla-Makarychev-Saberi-Teng ‘10:
Fix $s$ eigenvalues with $O(s)$ edges

$s = m / \log n \quad k = O(\log n)$
Preconditioning: solve in time $O(mk)$

Need $H \subseteq G$, $H$ is a tree + $m/k$ edges such that $\lambda_{max}(L_H^+ L_G) \leq k^2$

For low-stretch $T$,
At most $s$ eigenvalues larger than $O((m/s) \log n)$

Kolla-Makarychev-Saberi-Teng ‘10:
Fix $s$ eigenvalues with $O(s)$ edges

$s = m / \log n$ \hspace{1cm} $k = O(\log n)$

But, slow to choose edges
Preconditioning: solve in time $O(mk)$

Need $H \subseteq G$, $H$ is a tree + $m/k$ edges such that $\lambda_{max}(L_H^+L_G) \leq k^2$

For low-stretch $T$,
At most $s$ eigenvalues larger than $O((m/s) \log n)$

Koutis-Miller-Peng ‘10:

$k = O(\log^2 n)$

Easy to choose edges: sample by stretch
Faster Laplacian Solvers?

KMST ‘09 says $O(m \log n)$ might be possible

Very powerful primitive

Like sorting
Faster Laplacian Solvers?

KMST ‘09 says $O(m \log n)$ might be possible

Very powerful primitive

Like sorting

Better than BFS?
Uses for better graph exploration?
Faster local clustering?
Convergence of processes on graphs?
Conclusion

We’ve produced a lot of gems