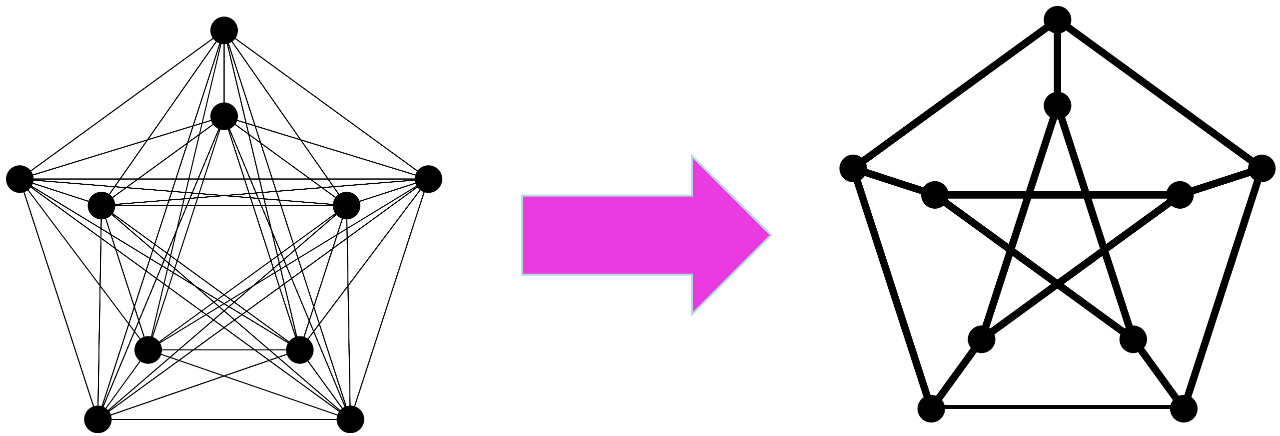
Laplacian Matrices of Graphs: Algorithms and Applications



Daniel A. Spielman
Dept. of Computer Science
Dept. of Statistics and Data Science
Yale Institute for Network Science

SIAM AN, 2017

Outline

Applications of Laplacian linear equations
- Interpolation on graphs
- Physical systems
- Optimization on graphs

Algorithms
- Sparsification
- Approximate Cholesky Factorization

Generalizations and recent developments

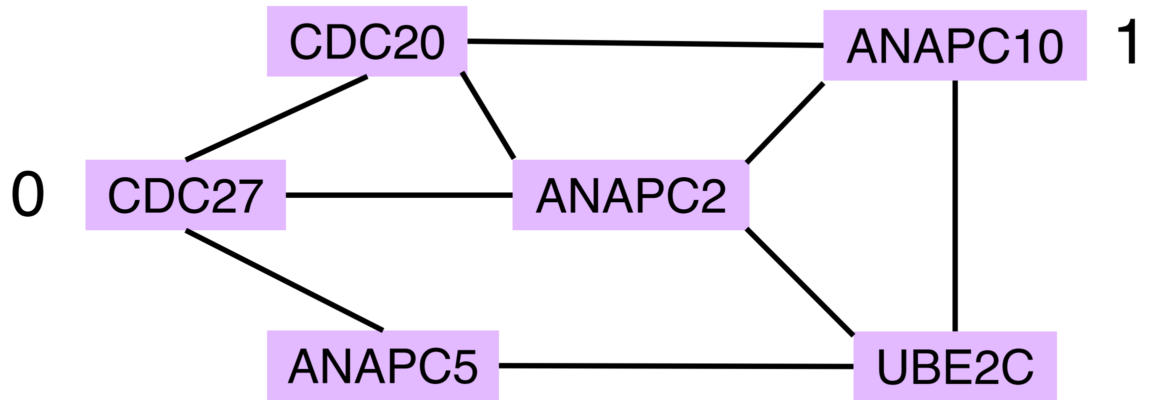
Interpolation on Graphs

(Zhu,Ghahramani,Lafferty '03)

Interpolate values of a function at all vertices from given values at a few vertices.

Minimize $$\sum_{(a,b)\in E} (x(a) - x(b))^2$$

Subject to given values



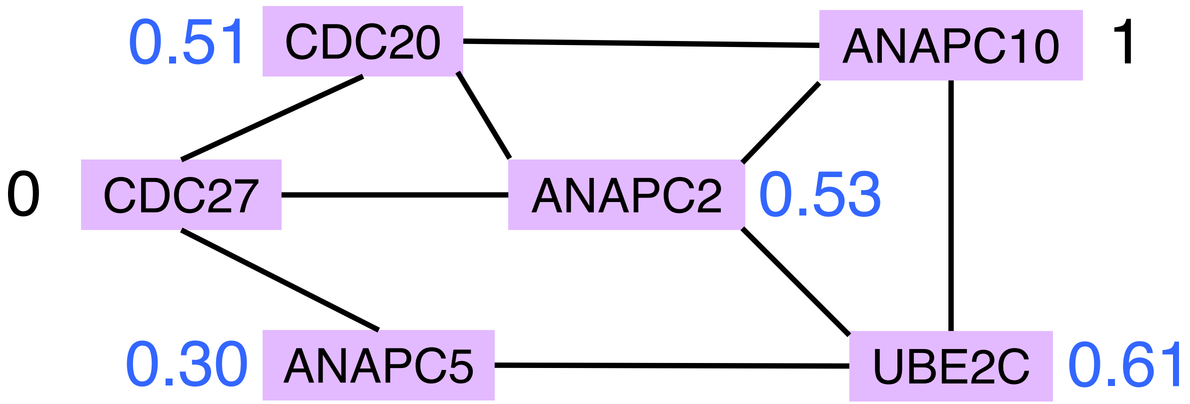
Interpolation on Graphs

(Zhu,Ghahramani,Lafferty '03)

Interpolate values of a function at all vertices from given values at a few vertices.

Minimize $$\sum_{(a,b)\in E} (x(a) - x(b))^2$$

Subject to given values



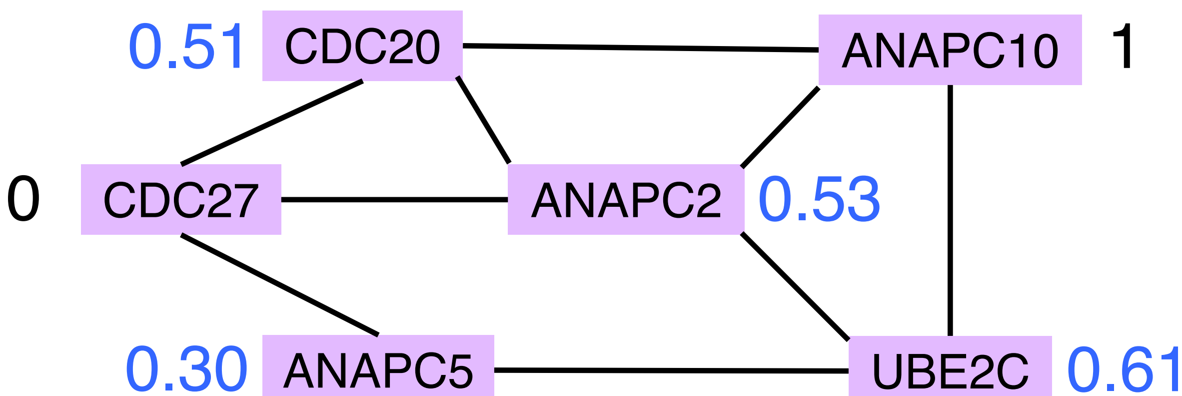
Interpolation on Graphs

(Zhu,Ghahramani,Lafferty '03)

Interpolate values of a function at all vertices from given values at a few vertices.

Minimize $$\sum_{(a,b)\in E} (x(a) - x(b))^2 = x^T L x$$

Subject to given values



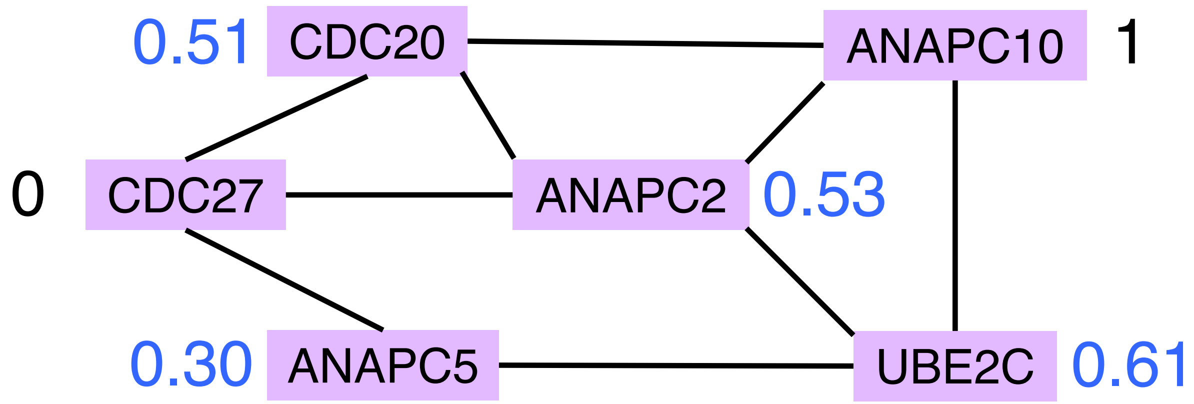
Interpolation on Graphs

(Zhu,Ghahramani,Lafferty '03)

Interpolate values of a function at all vertices from given values at a few vertices.

Minimize $$\sum_{(a,b)\in E} (x(a) - x(b))^2 = x^T L x$$

Subject to given values



Take derivatives. Minimize by solving Laplacian

The Laplacian Quadratic Form of a Graph

$$\sum_{(a,b)\in E} (x(a) - x(b))^2$$

The Laplacian Matrix of a Graph

$$x^T L_G x = \sum_{(a,b)\in E} (x(a) - x(b))^2$$

The Laplacian Matrix of a Weighted Graph

$$x^T L_G x = \sum_{(a,b) \in E} w_{a,b}(x(a) - x(b))^2$$

Positive real weights measuring

strength of connection

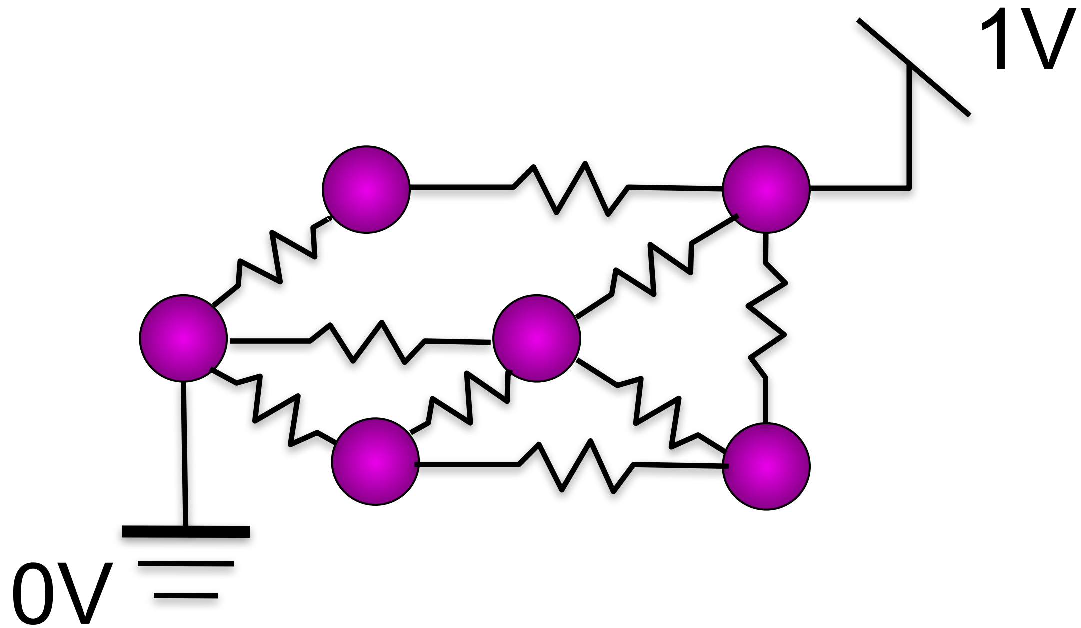
spring constant

1/resistance

Resistor Networks

View edges as resistors connecting vertices

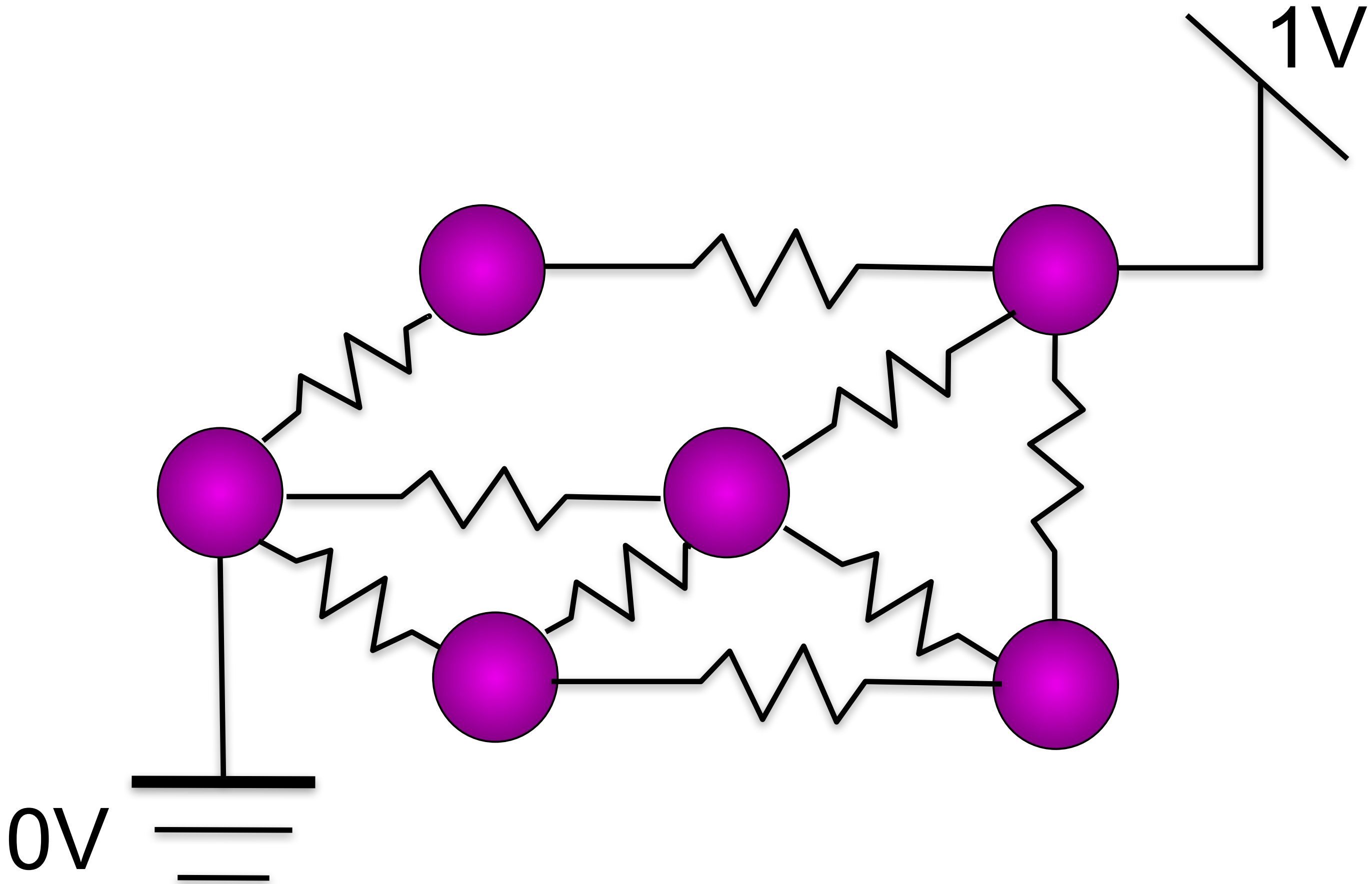
Apply voltages at some vertices.
Measure induced voltages and current flow.



Resistor Networks

Induced voltages minimize subject to constraints.

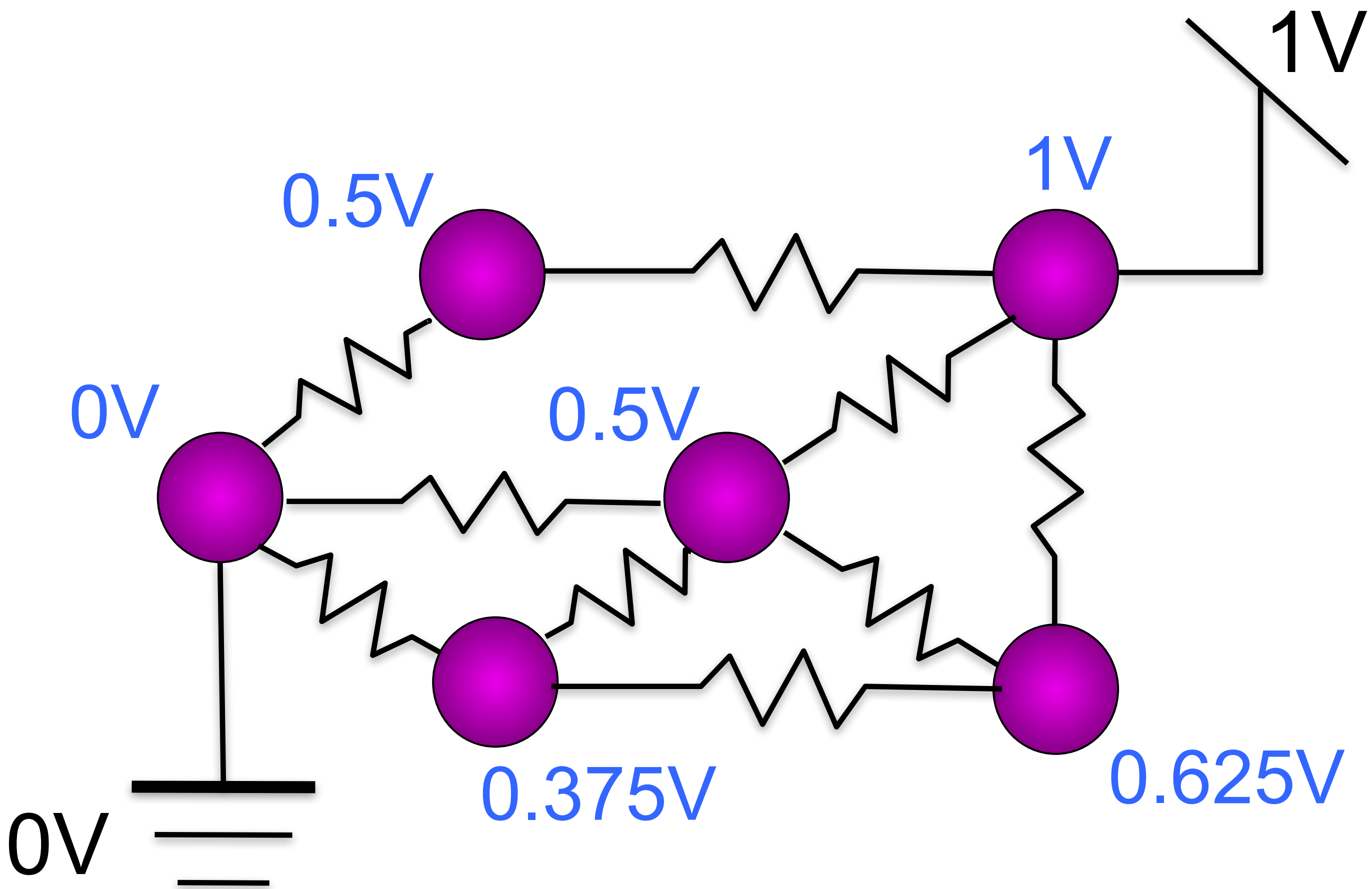
$$\sum_{(a,b)\in E} (x(a) - x(b))^2$$



Resistor Networks

Induced voltages minimize subject to constraints.

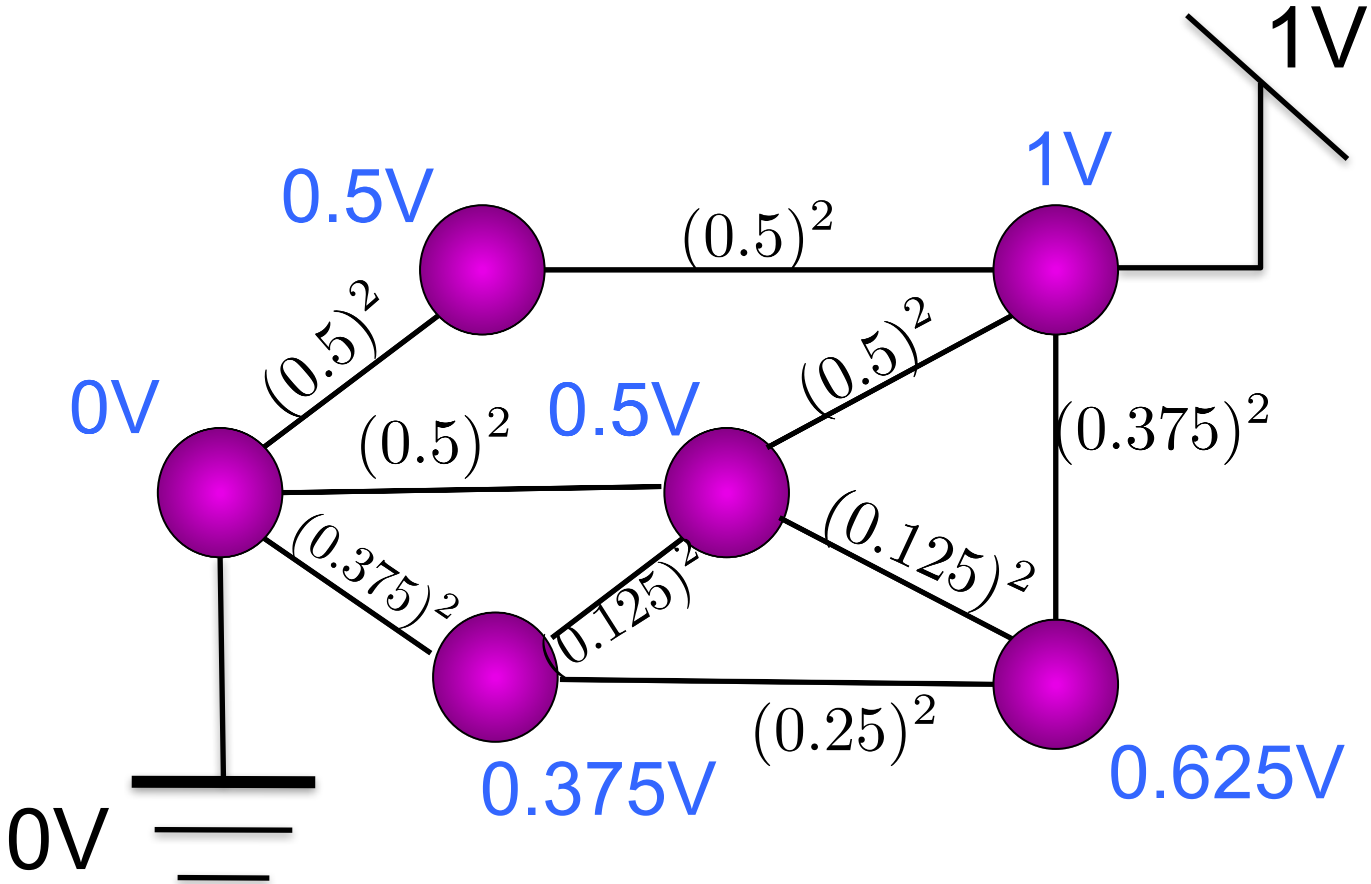
$$\sum_{(a,b)\in E} (x(a) - x(b))^2$$



Resistor Networks

Induced voltages minimize subject to constraints.

$$\sum_{(a,b)\in E} (x(a) - x(b))^2$$

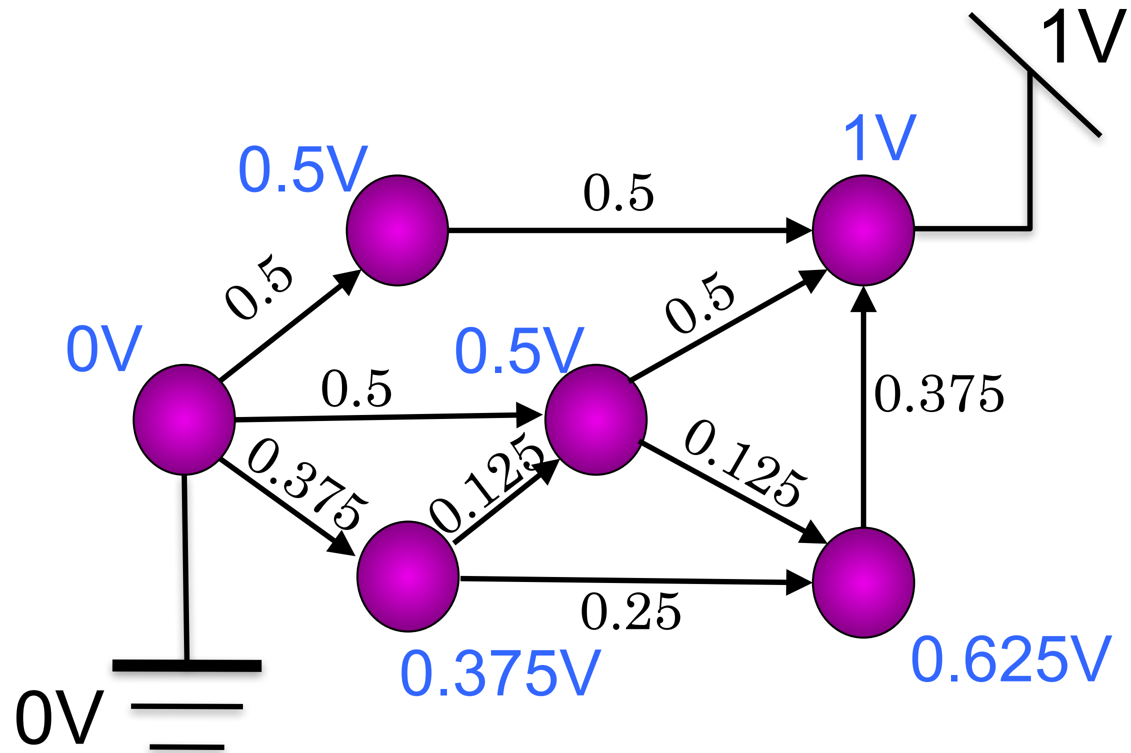


Resistor Networks

Induced voltages minimize subject to constraints.

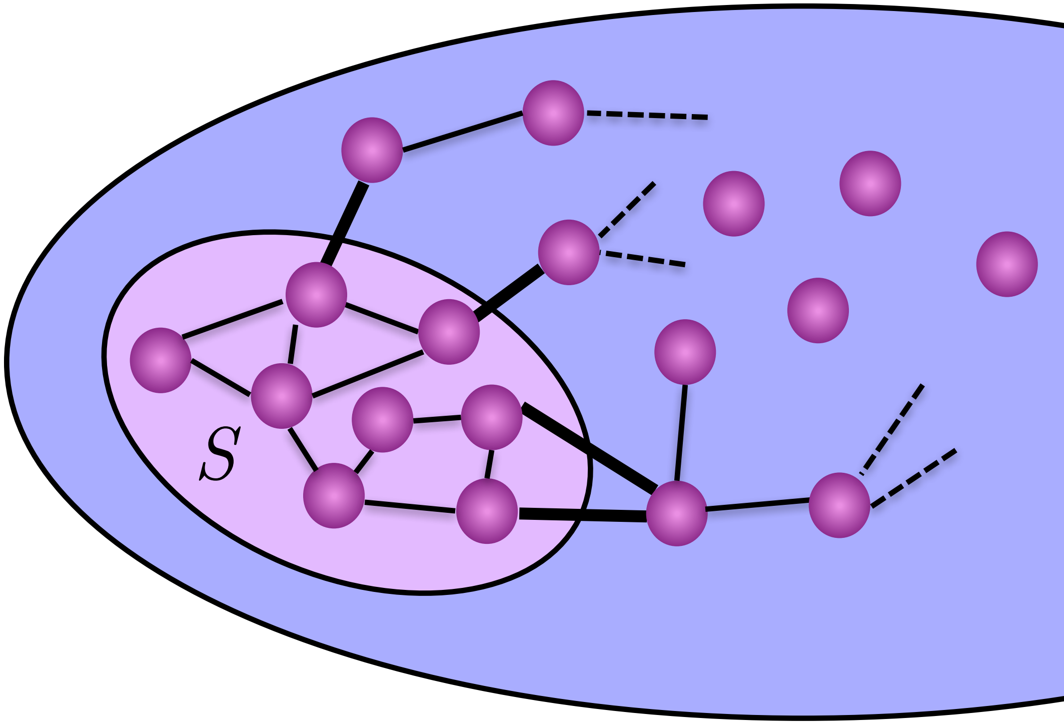
$$\sum_{(a,b)\in E} (x(a) - x(b))^2$$

Effective resistance = 1/(current flow at one volt)



Measuring boundaries of sets

Boundary: edges leaving a set

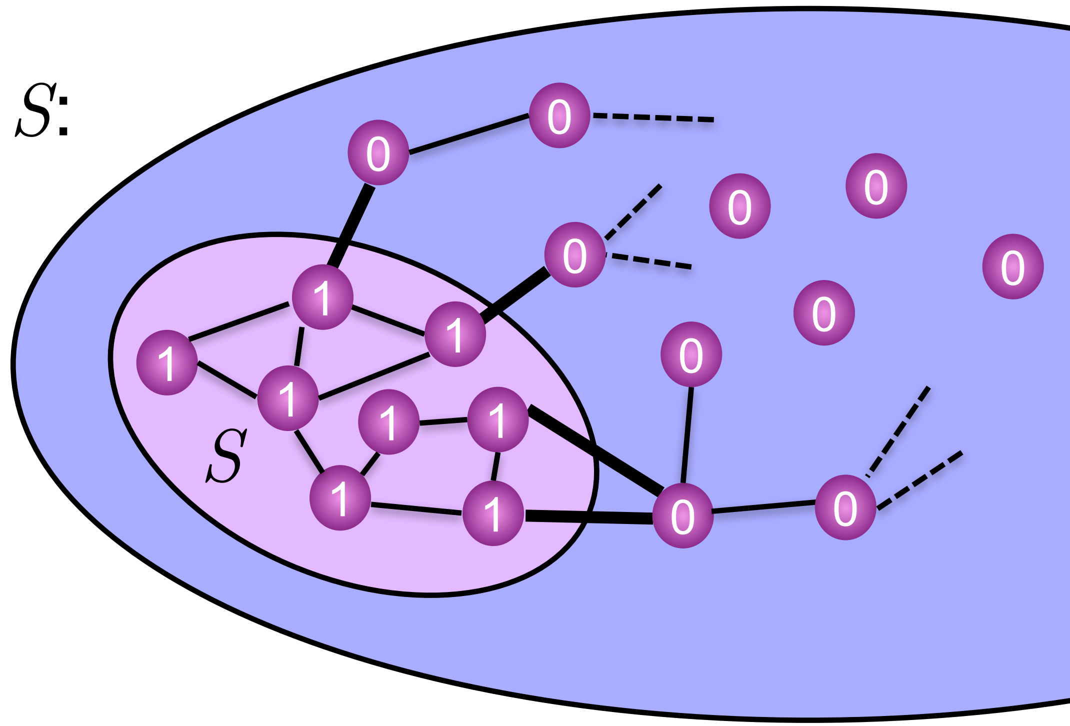


Measuring boundaries of sets

Boundary: edges leaving a set

Characteristic Vector of S:

$$x(a) = \begin{cases} 1 & a \text{ in } S \\ 0 & a \text{ not in } S \end{cases}$$

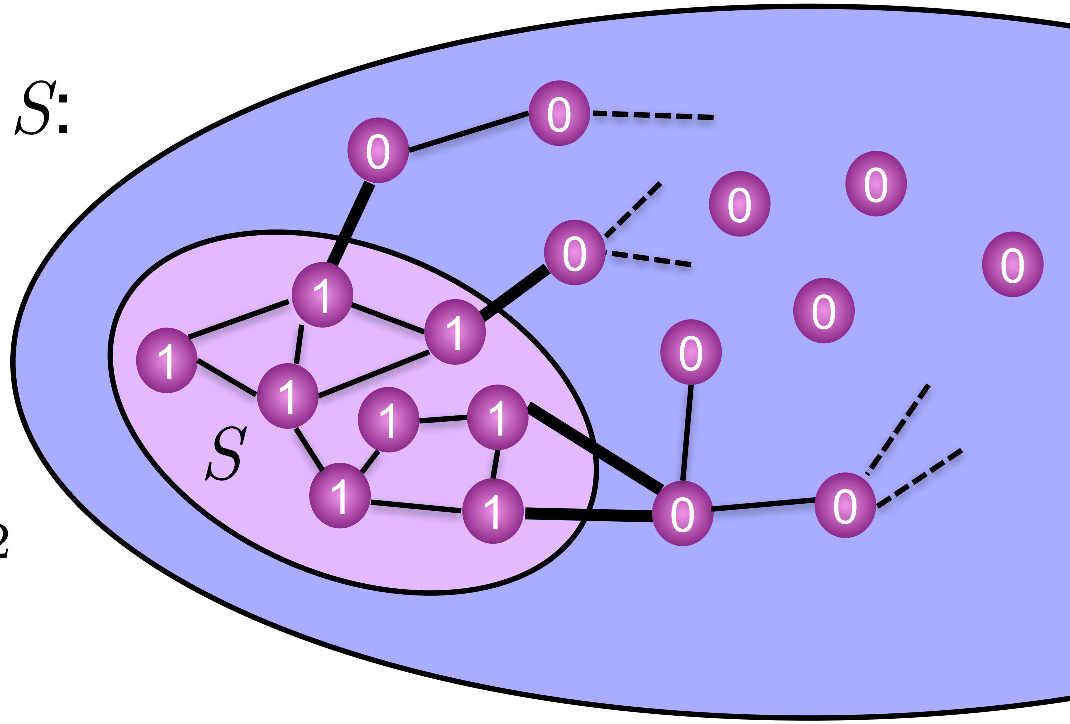


Measuring boundaries of sets

Boundary: edges leaving a set

Characteristic Vector of S:

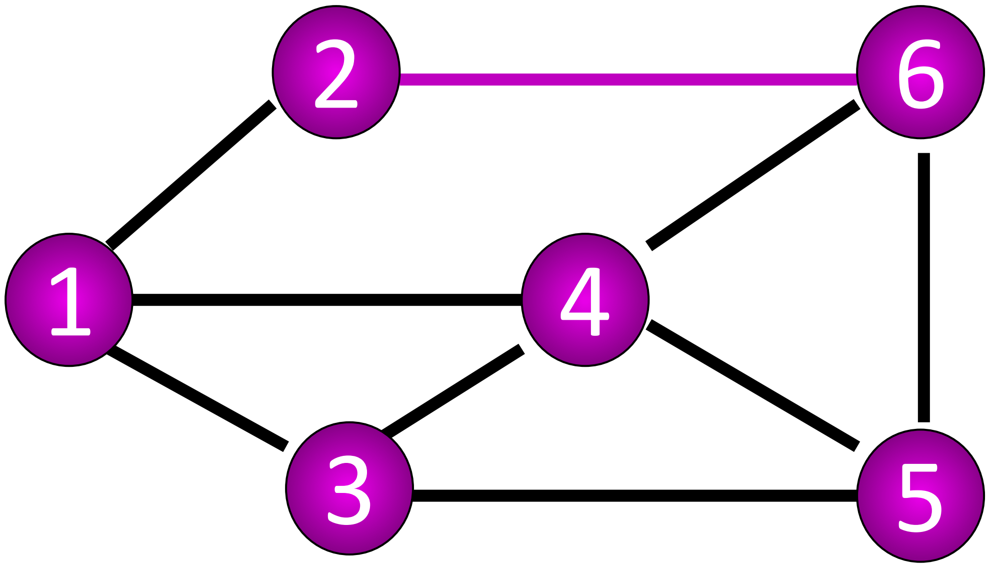
$$x(a) = \begin{cases} 1 & a \text{ in } S \\ 0 & a \text{ not in } S \end{cases}$$



$$\sum_{(a,b)\in E} (x(a) - x(b))^2$$

$$= |\text{boundary}(S)|$$

The Laplacian Matrix of a Graph



$$\begin{pmatrix} 3 & -1 & -1 & -1 & 0 & 0 \\ -1 & 2 & 0 & 0 & 0 & -1 \\ -1 & 0 & 3 & -1 & -1 & 0 \\ -1 & 0 & -1 & 4 & -1 & -1 \\ 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & -1 & 0 & -1 & -1 & 3 \end{pmatrix}$$

Symmetric

Non-positive off-diagonals

Diagonally dominant

The Laplacian Matrix of a Graph

$$x^T L_G x = \sum_{(a,b) \in E} w_{a,b} (x(a) - x(b))^2$$

$$L_G = \sum_{(a,b) \in E} w_{a,b} L_{a,b}$$

$$L_{1,2} = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 & -1 \end{pmatrix}$$

Quickly Solving Laplacian Equations

S,Teng '04: Using low-stretch trees and sparsifiers

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

Where m is number of non-zeros and n is dimension

Quickly Solving Laplacian Equations

S,Teng '04: Using low-stretch trees and sparsifiers

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

Koutis, Miller, Peng '11: Low-stretch trees and sampling

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

Where m is number of non-zeros and n is dimension

Quickly Solving Laplacian Equations

S,Teng '04: Using low-stretch trees and sparsifiers

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

Koutis, Miller, Peng '11: Low-stretch trees and sampling

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

Cohen, Kyng, Pachocki, Peng, Rao '14:

$$\widetilde{O}(m \log^{1/2} n \log \epsilon^{-1})$$

Where m is number of non-zeros and n is dimension

Quickly Solving Laplacian Equations

Good code:

LAMG (lean algebraic multigrid) – Livne-Brandt

CMG (combinatorial multigrid) – Koutis

approxChol in Laplacians.jl – S, Kyng-Sachdeva

Quickly Solving Laplacian Equations

S,Teng '04: Using low-stretch trees and sparsifiers

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

An ϵ-accurate solution to $L_G x = b$
is an $\widetilde{x}$ satisfying

$$\left\| \widetilde{x} - x \right\|_{L_G} \leq \epsilon \left\| x \right\|_{L_G}$$

where $\left\| v \right\|_{L_G} = \sqrt{v^T L_G v} = || L_G^{1/2} v ||$

Laplacians in Linear Programming

Laplacians appear when solving Linear Programs on
on graphs by Interior Point Methods

Maximum and Min-Cost Flow (Daitch, S ’08, Mądry ‘13)

Shortest Paths (Cohen, Mądry, Sankowski, Vladu ‘16)

Isotonic Regression (Kyng, Rao, Sachdeva ‘15)

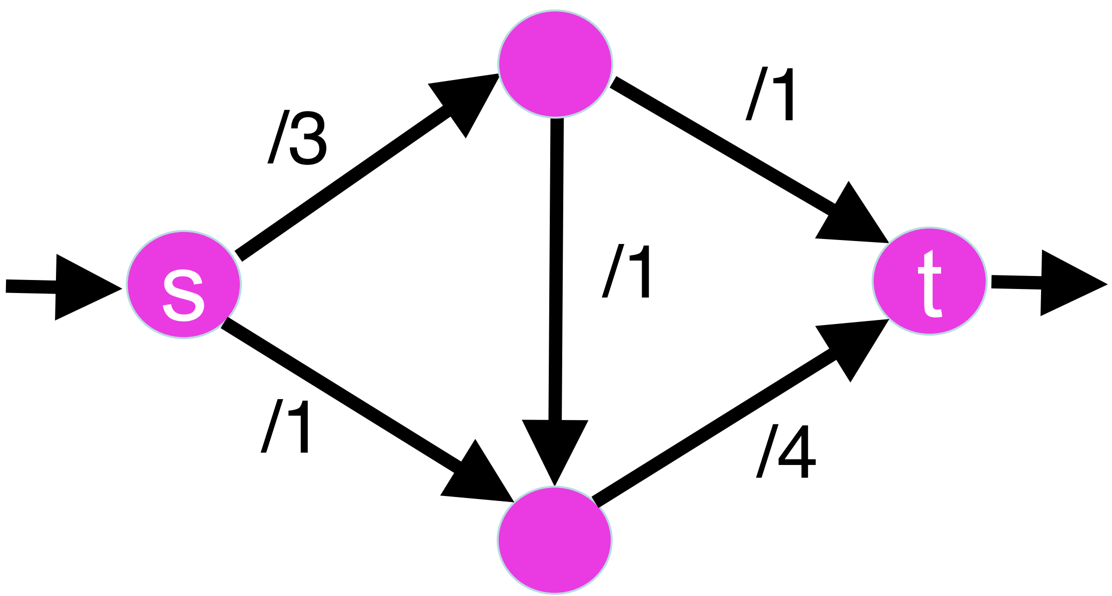
Lipschitz Learning : regularized interpolation on graphs
(Kyng, Rao, Sachdeva, S ‘15)

Interior Point Method for Maximum s-t Flow

maximize $f^{out}(s)$

subject to $\quad f^{out}(a) = f^{in}(a), \quad \forall a \notin \{s, t\}$

$$0 \leq f(a,b) \leq c(a,b), \quad \forall (a,b) \in E$$

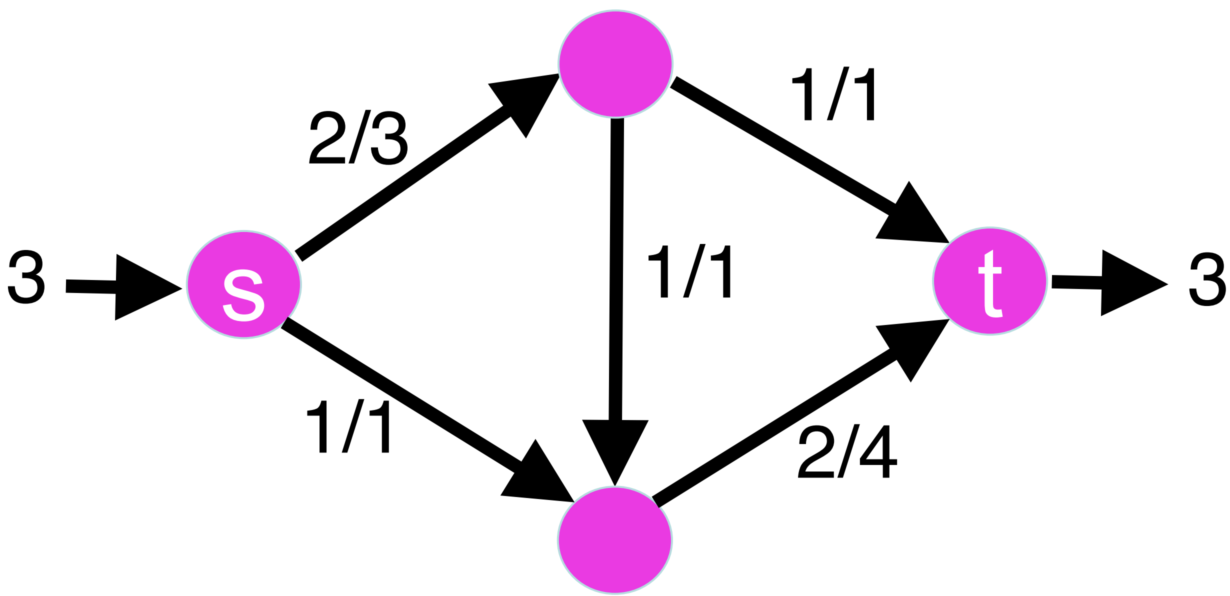


Interior Point Method for Maximum s-t Flow

maximize $f^{out}(s)$

subject to $f^{out}(a) = f^{in}(a), \quad \forall a \notin \{s, t\}$

$0 \leq f(a,b) \leq c(a,b), \quad \forall (a,b) \in E$



Interior Point Method for Maximum s-t Flow

maximize $f^{out}(s)$

subject to $f^{out}(a) = f^{in}(a), \quad \forall a \notin \{s, t\}$

$$0 \leq f(a,b) \leq c(a,b), \quad \forall (a,b) \in E$$

Multiple calls with varying weights $w_{a,b}$

minimize $\sum_{(a,b)\in E} w_{a,b} f(a,b)^2$

subject to $f^{out}(s) = f^{in}(t) = F$

$$f^{out}(a) = f^{in}(a), \quad \forall a \notin \{s, t\}$$

Interior Point Method for Min Cost Flow

minimize $\sum_{(a,b)} f(a,b)p(a,b)$

subject to $f^{out}(s) = f^{in}(t) = F$

$f^{out}(a) = f^{in}(a), \quad \forall a \notin \{s,t\}$

$0 \leq f(a,b) \leq c(a,b), \quad \forall (a,b) \in E$

Asymptotically fastest algorithms:

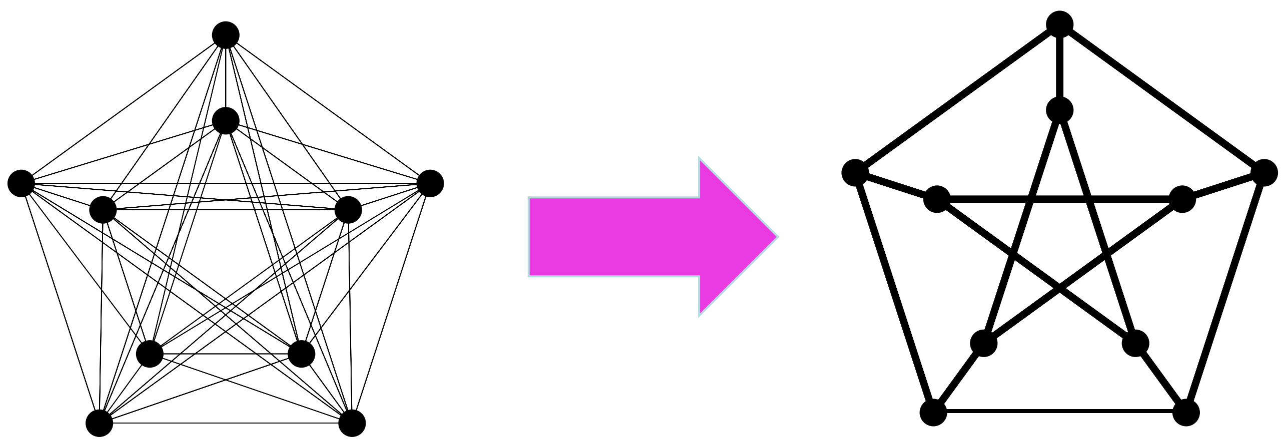
(Daitch, S '08; Mądry '13; Lee-Sidford '15)

Fastest on some large problems in practice?

(Fountoulakis, Rao, S '??)

Spectral Sparsification

Every graph can be approximated
by a sparse graph with a similar Laplacian



Approximating Graphs

A graph H is an ϵ-approximation of G if

for all x $$\frac{1}{1+\epsilon} \leq \frac{x^T L_H x}{x^T L_G x} \leq 1+\epsilon$$

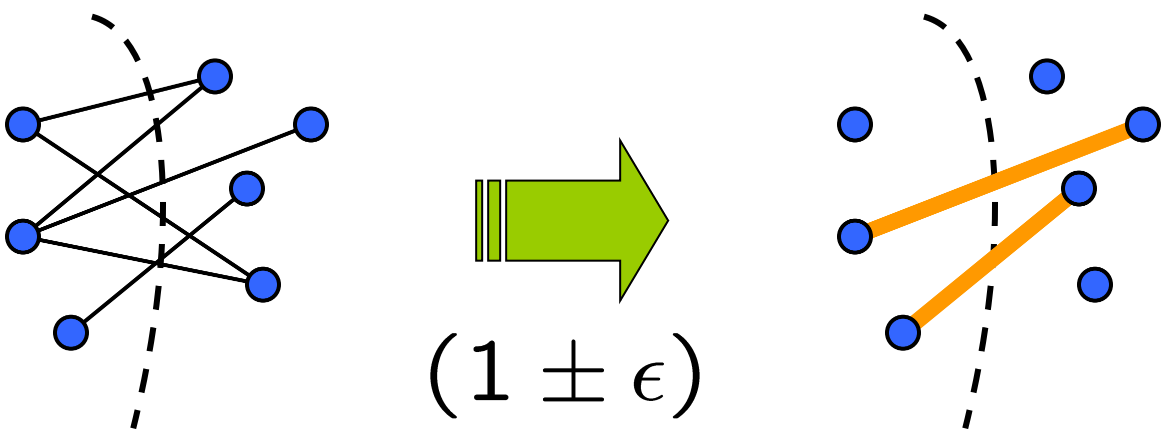
$$L_H \approx_\epsilon L_G$$

Approximating Graphs

A graph H is an ϵ-approximation of G if

for all x $$\frac{1}{1+\epsilon} \leq \frac{x^T L_H x}{x^T L_G x} \leq 1 + \epsilon$$

Preserves boundaries of every set



Approximating Graphs

A graph H is an ϵ-approximation of G if

for all x $$\frac{1}{1+\epsilon} \leq \frac{x^T L_H x}{x^T L_G x} \leq 1+\epsilon$$

Solutions to linear equations are similiar

$$L_H \approx_\epsilon L_G \iff L_H^{-1} \approx_\epsilon L_G^{-1}$$

Spectral Sparsification

(Batson, S, Srivastava '09)

Every graph G has an ϵ-approximation H with $n(2+\epsilon)^2/\epsilon^2$ edges

Spectral Sparsification

(Batson, S, Srivastava '09)

Every graph G has an ϵ-approximation H with $n(2+\epsilon)^2/\epsilon^2$ edges

Random regular graphs approximate complete graphs

Fast Spectral Sparsification

(S & Srivastava '08)

If sample each edge with probability inversely proportional to its effective resistance, only need $O(n \log n/\epsilon^2)$ samples

Takes time $O(m \log^2 n)$ (Koutis, Levin, Peng '12)

(Lee & Sun '17)

Can find an ϵ-approximation with $O(n/\epsilon^2)$ edges in nearly linear time.

Approximate Gaussian Elimination

(Kyng & Sachdeva '16)

Gaussian Elimination:
compute upper triangular U so that

$$L_G = U^T U$$

Approximate Gaussian Elimination:
compute sparse upper triangular U so that

$$L_G \approx U^T U$$

(See also Clarkson '03)

Additive view of Gaussian Elimination

Find U, upper triangular matrix, s.t $U^\top U = A$

$$A = \begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 5 & 0 & -1 \\ -8 & 0 & 14 & 0 \\ -4 & -1 & 0 & 7 \end{pmatrix}$$

Additive view of Gaussian Elimination

$$\begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 5 & 0 & -1 \\ -8 & 0 & 14 & 0 \\ -4 & -1 & 0 & 7 \end{pmatrix}$$

Find the rank-1 matrix that agrees on the first row and column.

$$\begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 1 & 2 & 1 \\ -8 & 2 & 4 & 2 \\ -4 & 1 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 4 \\ -1 \\ -2 \\ -1 \end{pmatrix} \begin{pmatrix} 4 \\ -1 \\ -2 \\ -1 \end{pmatrix}^\top$$

Additive view of Gaussian Elimination

$$\begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 5 & 0 & -1 \\ -8 & 0 & 14 & 0 \\ -4 & -1 & 0 & 7 \end{pmatrix} - \begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 1 & 2 & 1 \\ -8 & 2 & 4 & 2 \\ -4 & 1 & 2 & 1 \end{pmatrix}$$

Subtract the rank 1 matrix.

We have **eliminated the first variable.**

Additive view of Gaussian Elimination

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 4 & -2 & -2 \\ 0 & -2 & 10 & -2 \\ 0 & -2 & -2 & 6 \end{pmatrix}$$

Additive view of Gaussian Elimination

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 4 & -2 & -2 \\ 0 & -2 & 10 & -2 \\ 0 & -2 & -2 & 6 \end{pmatrix}$$

Find the rank-1 matrix that agrees on the **next** row and column.

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 4 & -2 & -2 \\ 0 & -2 & 1 & 1 \\ 0 & -2 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ -1 \\ -1 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ -1 \\ -1 \end{pmatrix}^\top$$

Additive view of Gaussian Elimination

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 4 & -2 & -2 \\ 0 & -2 & 10 & -2 \\ 0 & -2 & -2 & 6 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 4 & -2 & -2 \\ 0 & -2 & 1 & 1 \\ 0 & -2 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 9 & -3 \\ 0 & 0 & -3 & 5 \end{pmatrix}$$

Subtract the rank 1 matrix.

We have **eliminated the second variable.**

Additive view of Gaussian Elimination

$$A = \begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 5 & 0 & -1 \\ -8 & 0 & 14 & 0 \\ -4 & -1 & 0 & 7 \end{pmatrix}$$

$$= \begin{pmatrix} 4 \\ -1 \\ -2 \\ -1 \end{pmatrix} \begin{pmatrix} 4 \\ -1 \\ -2 \\ -1 \end{pmatrix}^\top + \begin{pmatrix} 0 \\ 2 \\ -1 \\ -1 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ -1 \\ -1 \end{pmatrix}^\top + \begin{pmatrix} 0 \\ 0 \\ 3 \\ -1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 3 \\ -1 \end{pmatrix}^\top + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 2 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 2 \end{pmatrix}^\top$$

Running time proportional to sum of squares
of number of non-zeros in these vectors.

Additive view of Gaussian Elimination

$$A = \begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 5 & 0 & -1 \\ -8 & 0 & 14 & 0 \\ -4 & -1 & 0 & 7 \end{pmatrix}$$

$$= \begin{pmatrix} 4 \\ -1 \\ -2 \\ -1 \end{pmatrix} \begin{pmatrix} 4 \\ -1 \\ -2 \\ -1 \end{pmatrix}^\top + \begin{pmatrix} 0 \\ 2 \\ -1 \\ -1 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ -1 \\ -1 \end{pmatrix}^\top + \begin{pmatrix} 0 \\ 0 \\ 3 \\ -1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 3 \\ -1 \end{pmatrix}^\top + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 2 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 2 \end{pmatrix}^\top$$

$$= \begin{pmatrix} 4 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ -2 & -1 & 3 & 0 \\ -1 & -1 & -1 & 2 \end{pmatrix} \begin{pmatrix} 4 & -1 & -2 & -1 \\ 0 & 2 & -1 & -1 \\ 0 & 0 & 3 & -1 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

Additive view of Gaussian Elimination

$$A = \begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 5 & 0 & -1 \\ -8 & 0 & 14 & 0 \\ -4 & -1 & 0 & 7 \end{pmatrix}$$

$$= \begin{pmatrix} 4 \\ -1 \\ -2 \\ -1 \end{pmatrix} \begin{pmatrix} 4 \\ -1 \\ -2 \\ -1 \end{pmatrix}^\top + \begin{pmatrix} 0 \\ 2 \\ -1 \\ -1 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ -1 \\ -1 \end{pmatrix}^\top + \begin{pmatrix} 0 \\ 0 \\ 3 \\ -1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 3 \\ -1 \end{pmatrix}^\top + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 2 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 2 \end{pmatrix}^\top$$

$$= \begin{pmatrix} 4 & -1 & -2 & -1 \\ 0 & 2 & -1 & -1 \\ 0 & 0 & 3 & -1 \\ 0 & 0 & 0 & 2 \end{pmatrix}^\top \begin{pmatrix} 4 & -1 & -2 & -1 \\ 0 & 2 & -1 & -1 \\ 0 & 0 & 3 & -1 \\ 0 & 0 & 0 & 2 \end{pmatrix} = U^\top U$$

Gaussian Elimination of Laplacians

If this is a Laplacian, then so is this

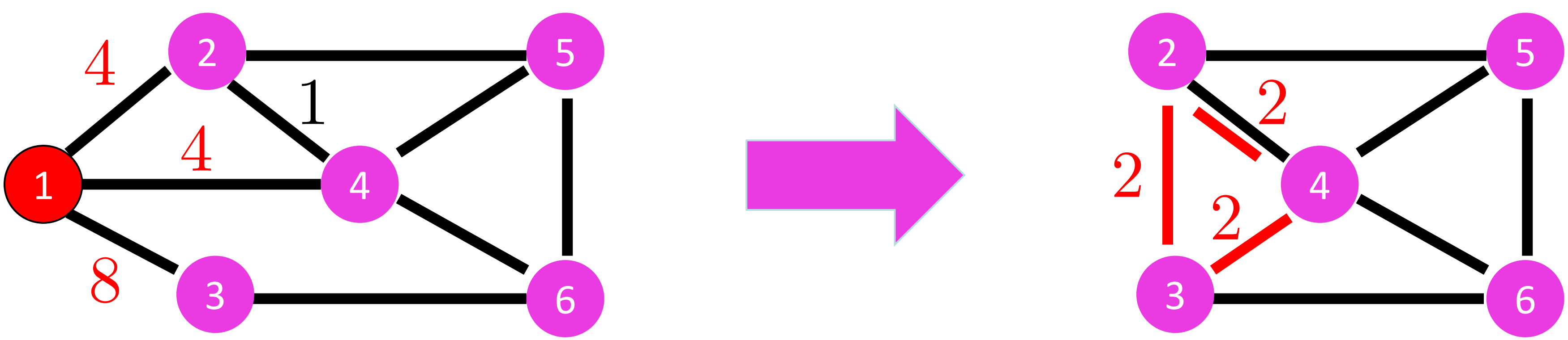
$$\begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 5 & 0 & -1 \\ -8 & 0 & 14 & 0 \\ -4 & -1 & 0 & 7 \end{pmatrix} - \begin{pmatrix} 4 \\ -1 \\ -2 \\ -1 \end{pmatrix} \begin{pmatrix} 4 \\ -1 \\ -2 \\ -1 \end{pmatrix}^T = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 4 & -2 & -2 \\ 0 & -2 & 10 & -2 \\ 0 & -2 & -2 & 6 \end{pmatrix}$$

Gaussian Elimination of Laplacians

If this is a Laplacian, then so is this

$$\begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 5 & 0 & -1 \\ -8 & 0 & 14 & 0 \\ -4 & -1 & 0 & 7 \end{pmatrix} - \begin{pmatrix} 4 \\ -1 \\ -2 \\ -1 \end{pmatrix} \begin{pmatrix} 4 \\ -1 \\ -2 \\ -1 \end{pmatrix}^T = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 4 & -2 & -2 \\ 0 & -2 & 10 & -2 \\ 0 & -2 & -2 & 6 \end{pmatrix}$$

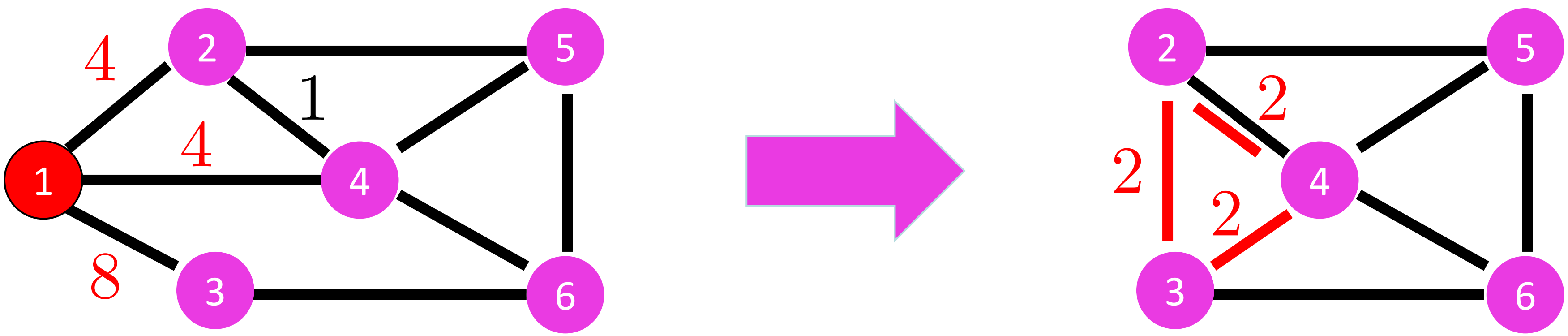
When eliminate a node, add a clique on its neighbors



Approximate Gaussian Elimination

(Kyng & Sachdeva '16)

1. when eliminate a node, add a clique on its neighbors



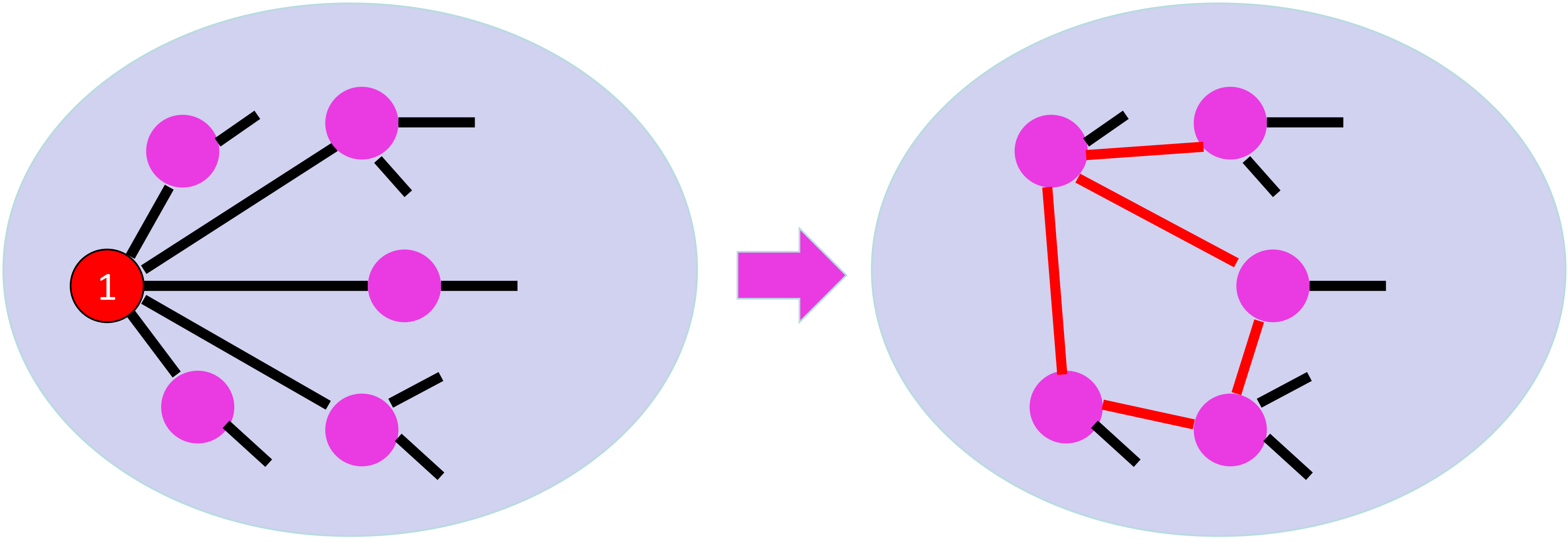
2. Sparsify that clique, without ever constructing it

Approximate Gaussian Elimination

(Kyng & Sachdeva '16)

When eliminate a node of degree d,

add d edges at random between its neighbors, sampled with probability proportional to the weight of the edge to the eliminated node



Approximate Gaussian Elimination

(Kyng & Sachdeva '16)

1. Initialize by randomly ordering the vertices,

2. and making $O(\log^2 n)$ copies of every edge

Total time is $O(m \log^3 n)$

Approximate Gaussian Elimination

(Kyng & Sachdeva ‘16)

Analysis by Random Matrix Theory:

Write $U^T U$ as a sum of random matrices.

$$\mathbb{E}\left[U^T U\right] = L_G$$

Random permutation and copying
control the variances of the random matrices

Apply Matrix Freedman inequality (Tropp ‘11)

Approximate Gaussian Elimination

(Kyng & Sachdeva '16)

1. Initialize by randomly ordering the vertices,

2. and making $O(\log^2 n)$ copies of every edge

Total time is $O(m \log^3 n)$

Can improve asymptotics by sacrificing some simplicity

Approximate Gaussian Elimination

(Kyng & Sachdeva '16)

1. Initialize by randomly ordering the vertices,

2. and making $O(\log^2 n)$ copies of every edge

Total time is $O(m \log^3 n)$

Can improve asymptotics by sacrificing some simplicity

Can improve practice by sacrificing some theory

Approximate Gaussian Elimination

A fast implementation in Laplacians.jl

Usually 400k-1M edges per second, for 8 digits

Competes with LAMG, CMG, incomplete Cholesky.

Never much slower.

Sometimes much faster.

Recent Developments

Other families of linear systems

(Kyng, Lee, Peng, Sachdeva, S '16)

complex-weighted Laplacians $\begin{pmatrix} 1 & e^{i\theta} \\ e^{-i\theta} & 1 \end{pmatrix}$

connection Laplacians $\begin{pmatrix} I & Q \\ Q^T & I \end{pmatrix}$

Recent Developments

Laplacians of Directed Graphs!

(Cohen, Kelner, Peebles, Peng, Sidford, Vladu ‘16)
(Cohen, Kelner, Peebles, Peng, Rao, Sidford, Vladu ‘16)
+1 to come with Rasmus Kyng (see his thesis)

With analyses of iterative methods for non-symmetric systems.

Fast computation of stable distribution of random walks.

Recent Developments

Laplacians.jl

Laplacian equation solvers
Sparsification
Low-stretch spanning trees
Interior point methods
Local graph clustering
Tricky graph generators

To learn more

My web page on:

Laplacian linear equations, sparsification, local graph clustering, low-stretch spanning trees, and so on.

My class notes from

“Graphs and Networks” and “Spectral Graph Theory”

Theses of

Richard Peng, Aaron Sidford, Yin Tat Lee, and Rasmus Kyng

$Lx = b$, by Nisheeth Vishnoi