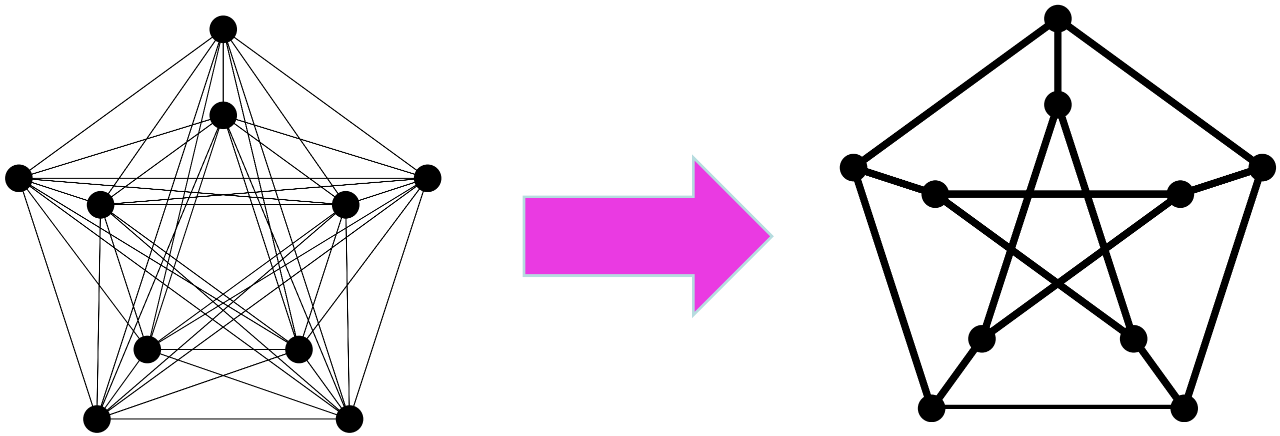
Laplacian Matrices of Graphs: Algorithms and Applications



Daniel A. Spielman

Yale Institute For Network Science

ICML, June 21, 2016

Outline

- Laplacians
  - Interpolation on graphs
  - Spring networks
  - Clustering
  - Isotonic regression
- Sparsification
- Solving Laplacian Equations
  - Best results
  - The simplest algorithm

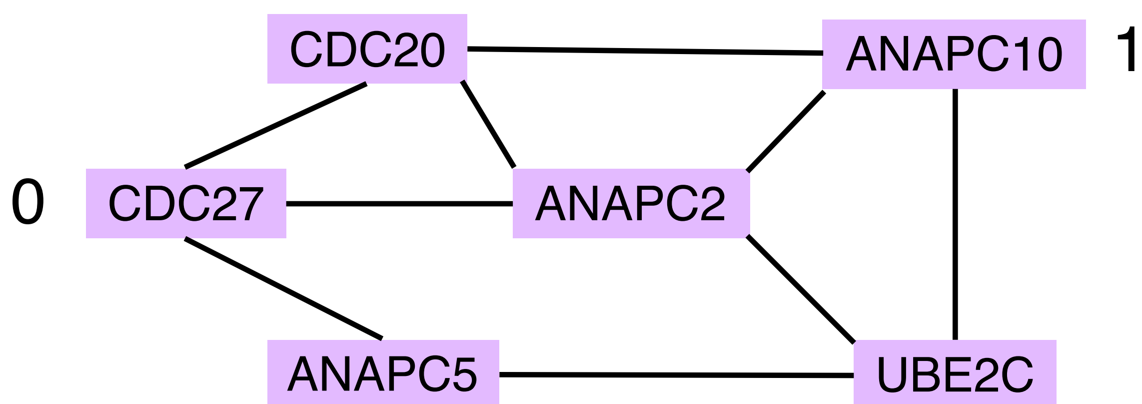
Interpolation on Graphs

(Zhu,Ghahramani,Lafferty '03)

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

Minimize $$\sum_{(i,j) \in E} (x(i) - x(j))^2$$

Subject to given values



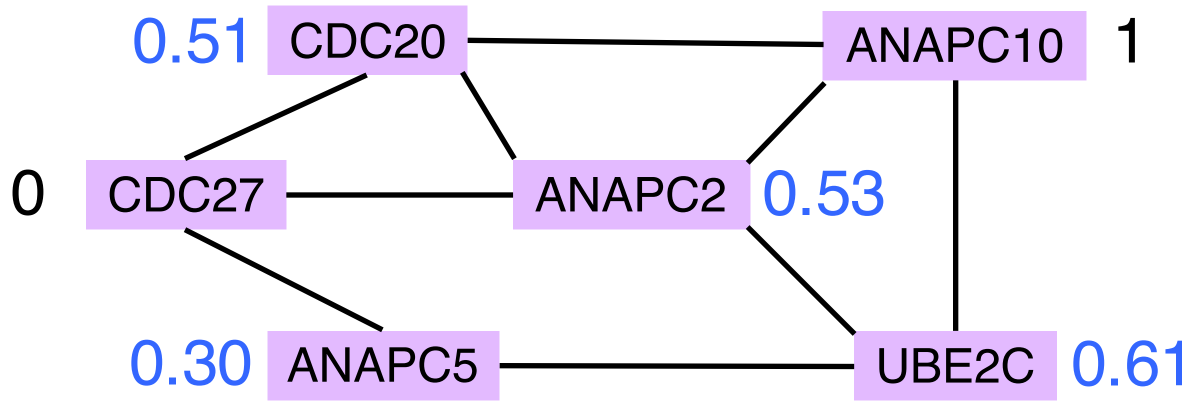
Interpolation on Graphs

(Zhu,Ghahramani,Lafferty '03)

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

Minimize $$\sum_{(i,j)\in E} (x(i) - x(j))^2$$

Subject to given values



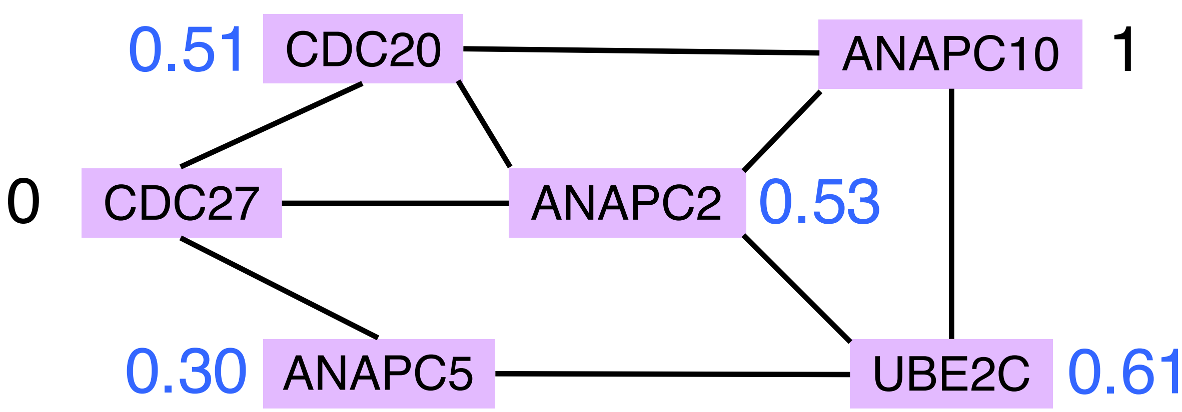
Interpolation on Graphs

(Zhu,Ghahramani,Lafferty '03)

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

Minimize $$\sum_{(i,j)\in E} (x(i) - x(j))^2 = x^T L_G x$$

Subject to given values



Take derivatives. Minimize by solving Laplacian

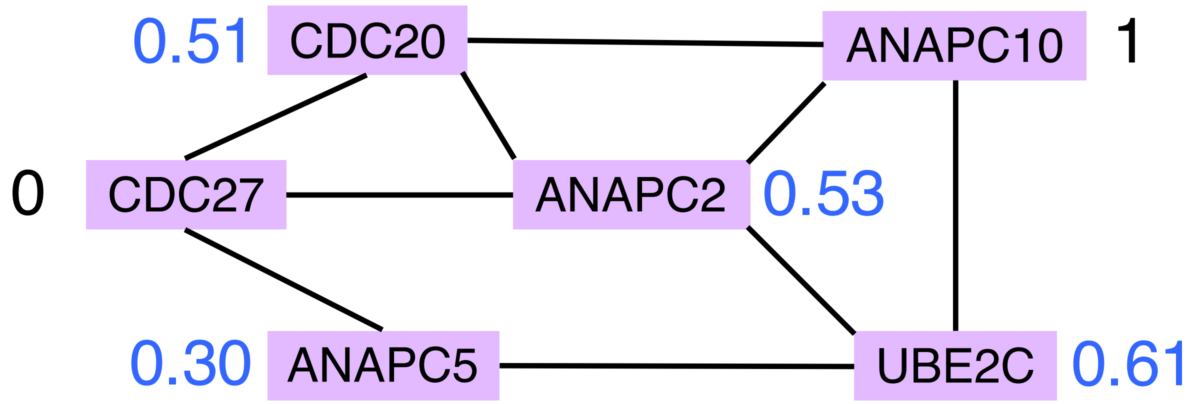
Interpolation on Graphs

(Zhu,Ghahramani,Lafferty '03)

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

Minimize $$\sum_{(i,j)\in E} (x(i) - x(j))^2 = x^T L_G x$$

Subject to given values



The Laplacian Quadratic Form

$$\sum_{(i,j)\in E} (x(i) - x(j))^2$$

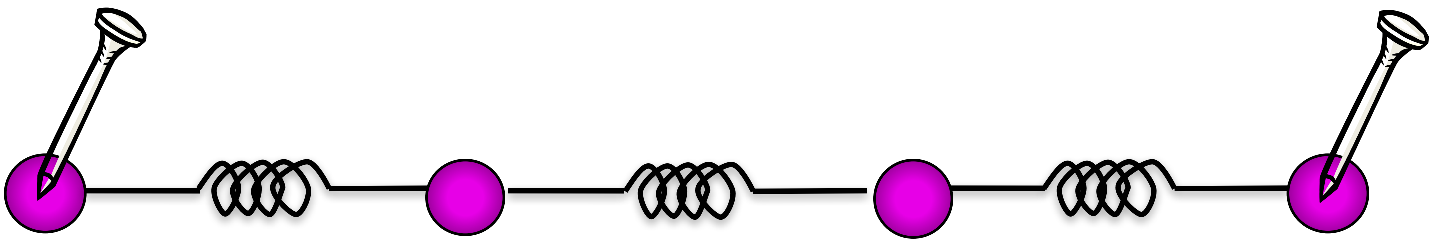
The Laplacian Matrix of a Graph

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

Spring Networks

View edges as rubber bands or ideal linear springs

Nail down some vertices, let rest settle

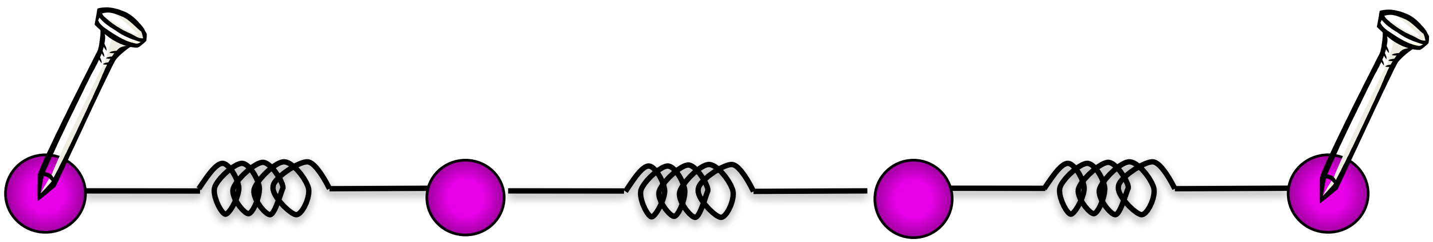


In equilibrium, nodes are averages of neighbors.

Spring Networks

View edges as rubber bands or ideal linear springs

Nail down some vertices, let rest settle

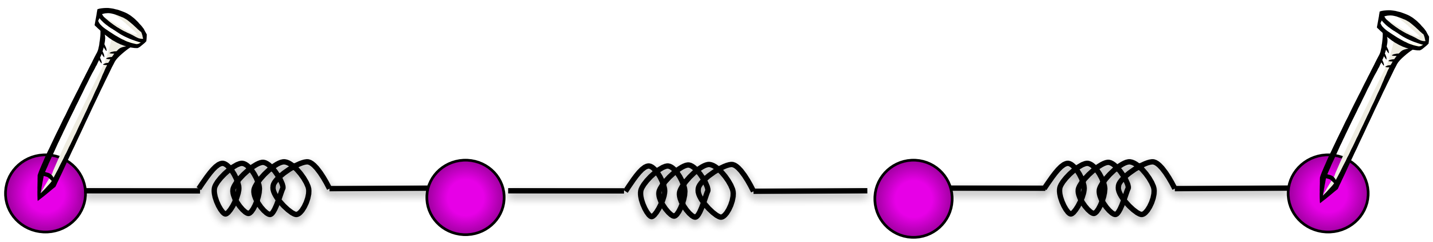


When stretched to length ℓ

potential energy is $\ell^2/2$

Spring Networks

Nail down some vertices, let rest settle



Physics: position minimizes total potential energy

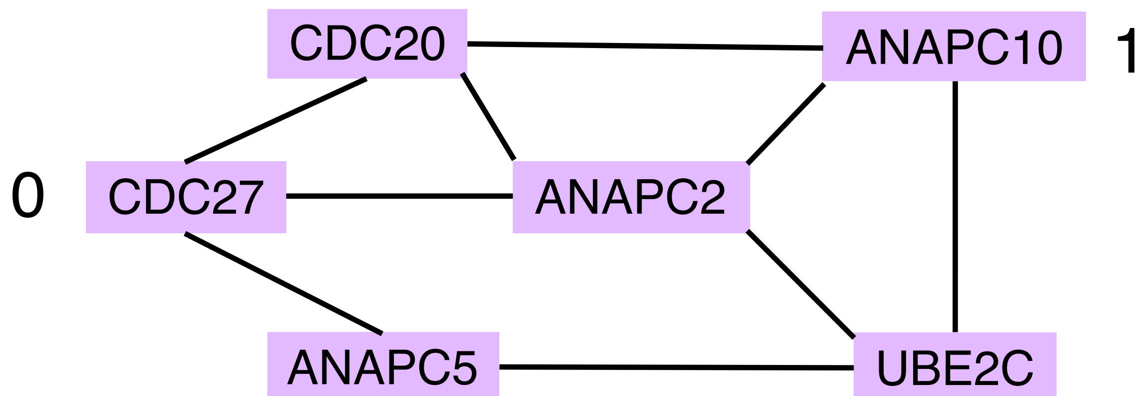
$$\frac{1}{2} \sum_{(i,j) \in E} (x(i) - x(j))^2$$

subject to boundary constraints (nails)

Spring Networks

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

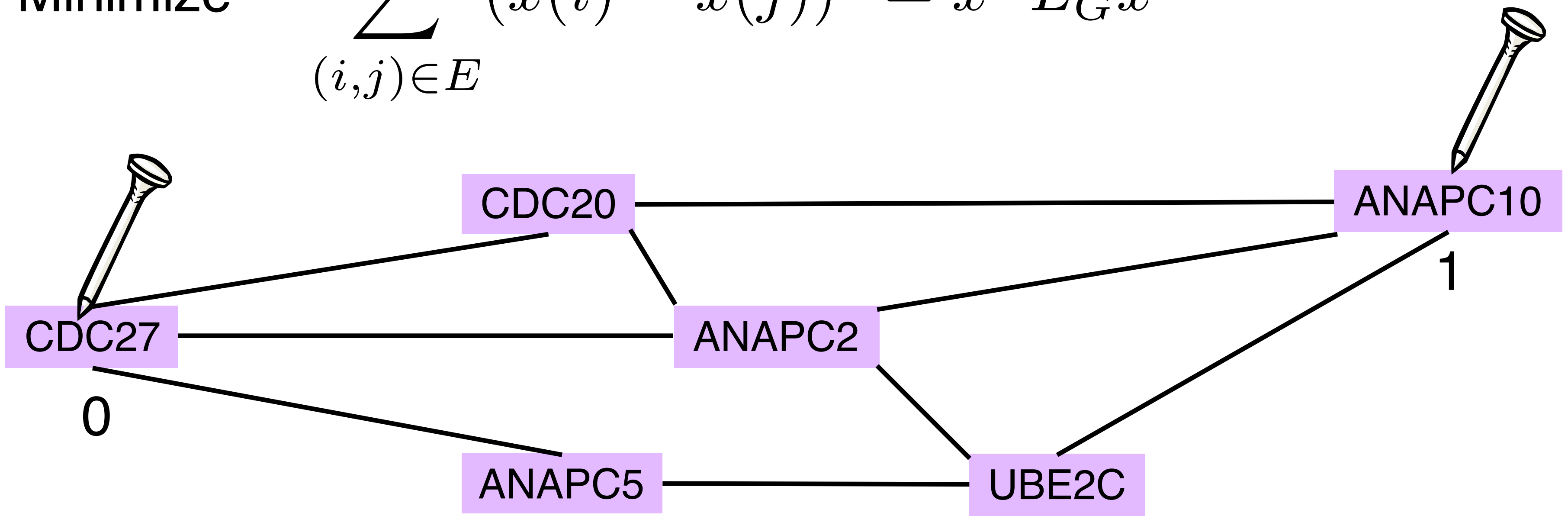
Minimize $$\sum_{(i,j)\in E} (x(i) - x(j))^2 = x^T L_G x$$



Spring Networks

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

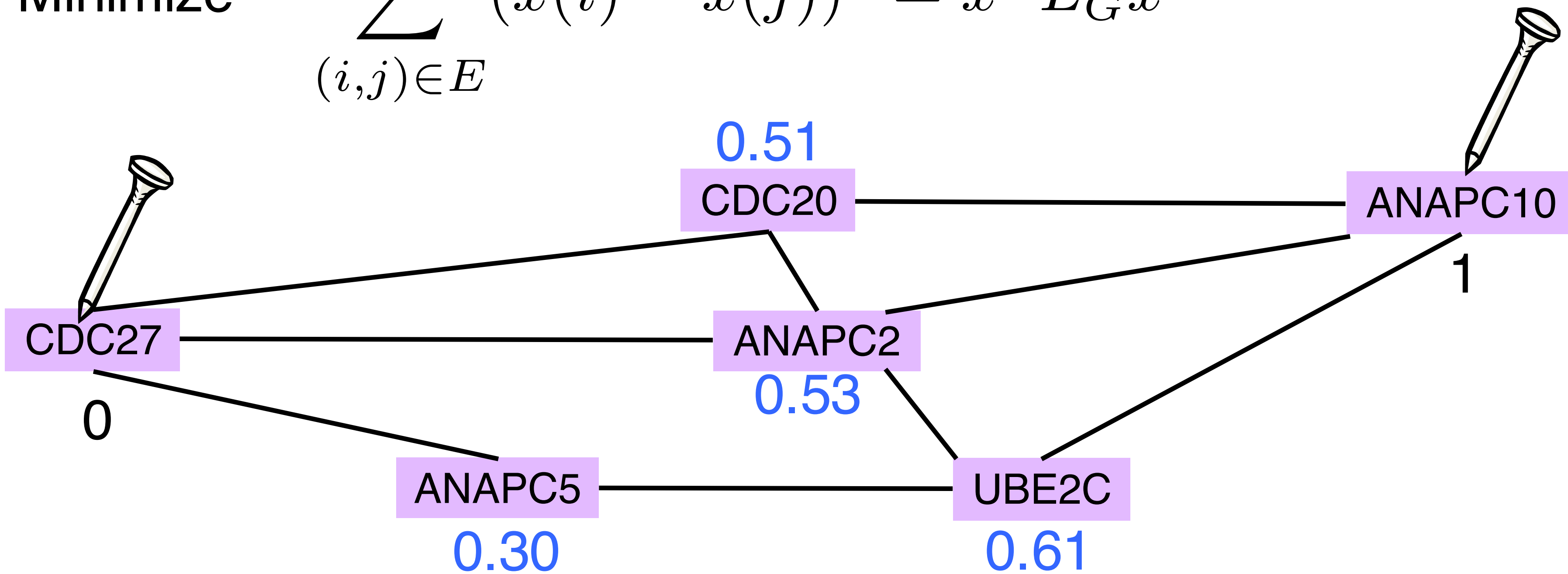
Minimize $$\sum_{(i,j)\in E} (x(i) - x(j))^2 = x^T L_G x$$



Spring Networks

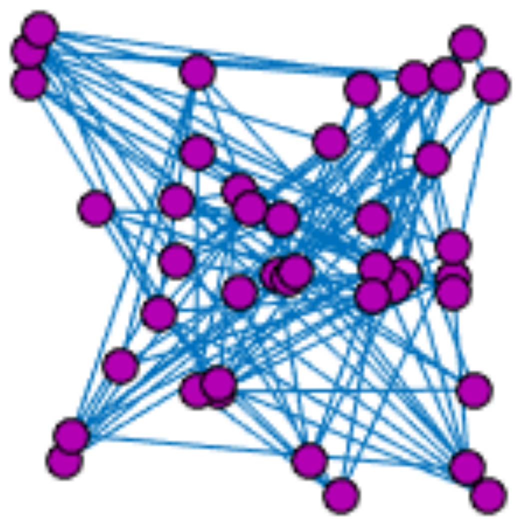
Interpolate values of a function at all vertices from given values at a few vertices.

Minimize $$\sum_{(i,j)\in E} (x(i) - x(j))^2 = x^T L_G x$$

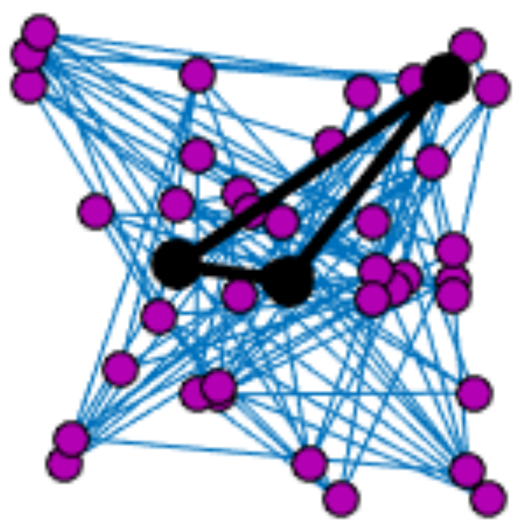


In the solution, variables are the average of their neighbors

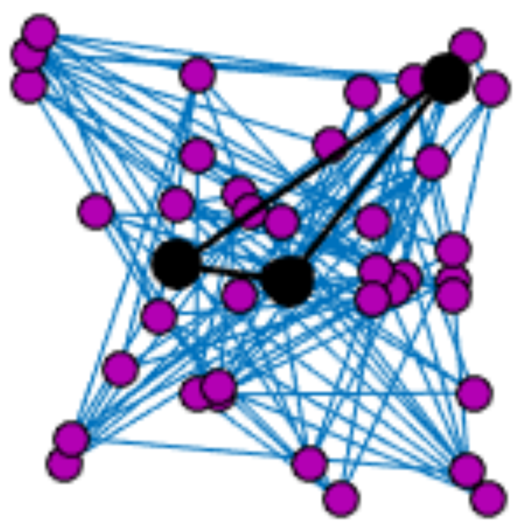
Drawing by Spring Networks (Tutte '63)



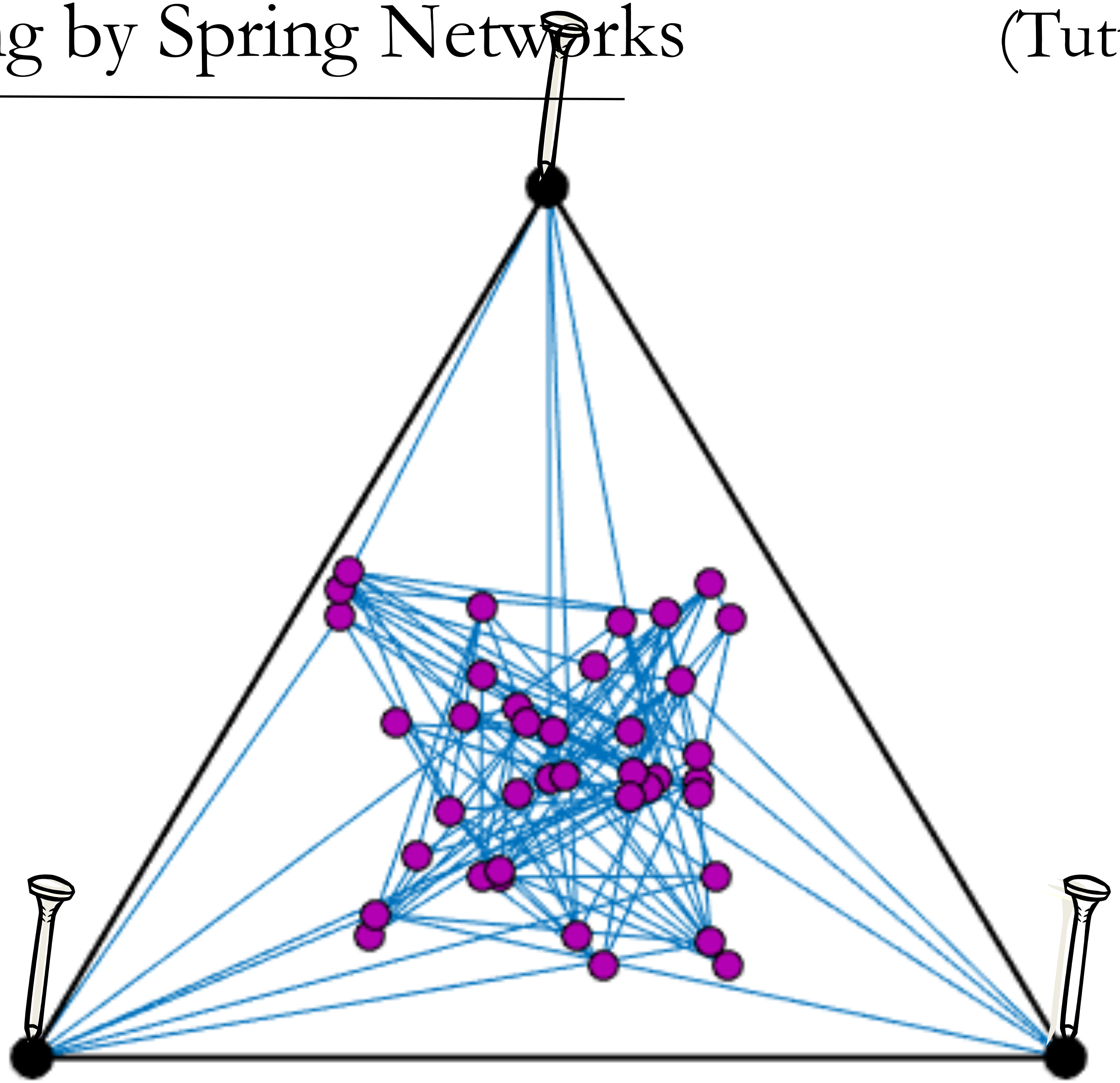
Drawing by Spring Networks (Tutte '63)



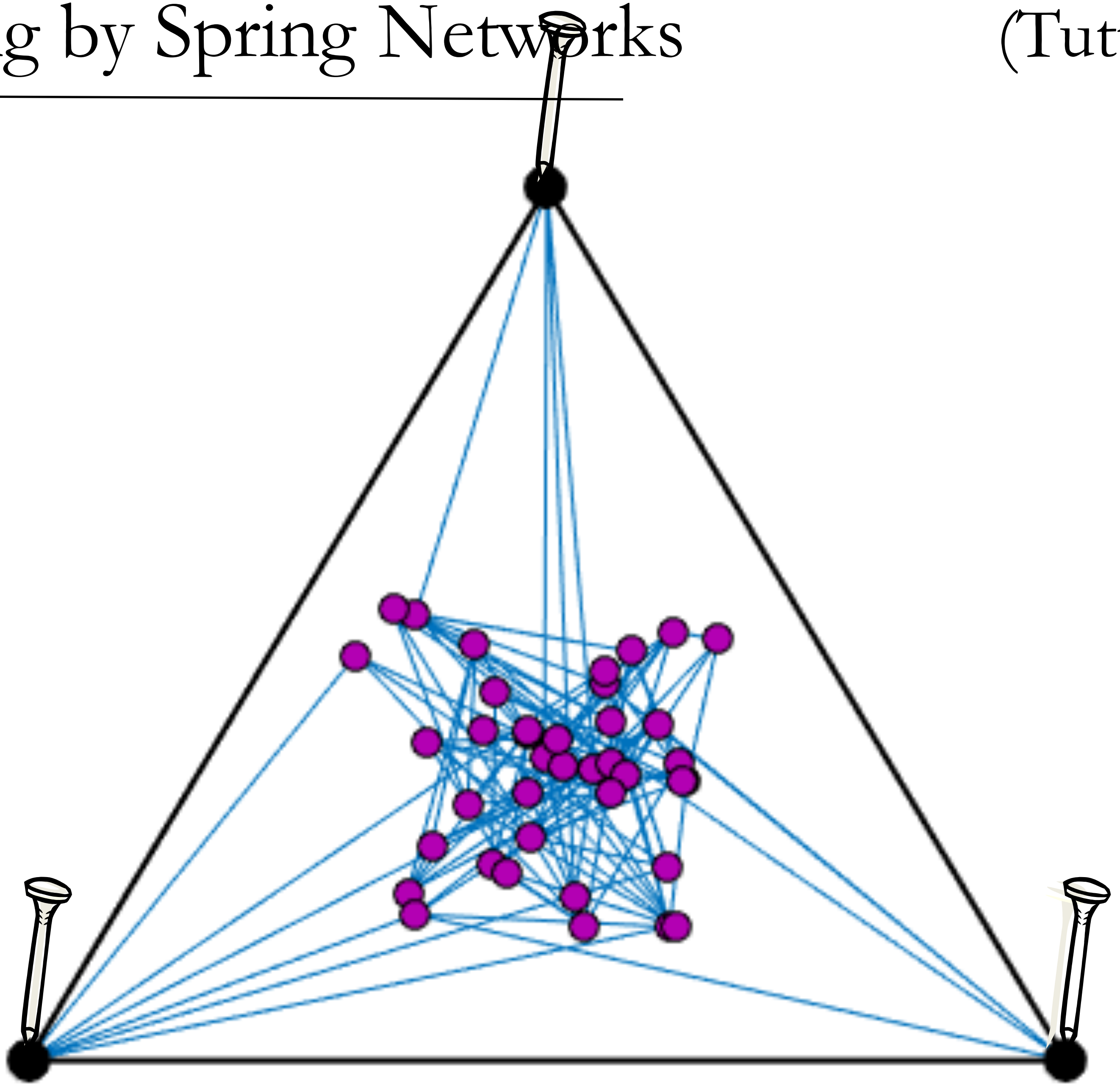
Drawing by Spring Networks (Tutte '63)



Drawing by Spring Networks (Tutte '63)

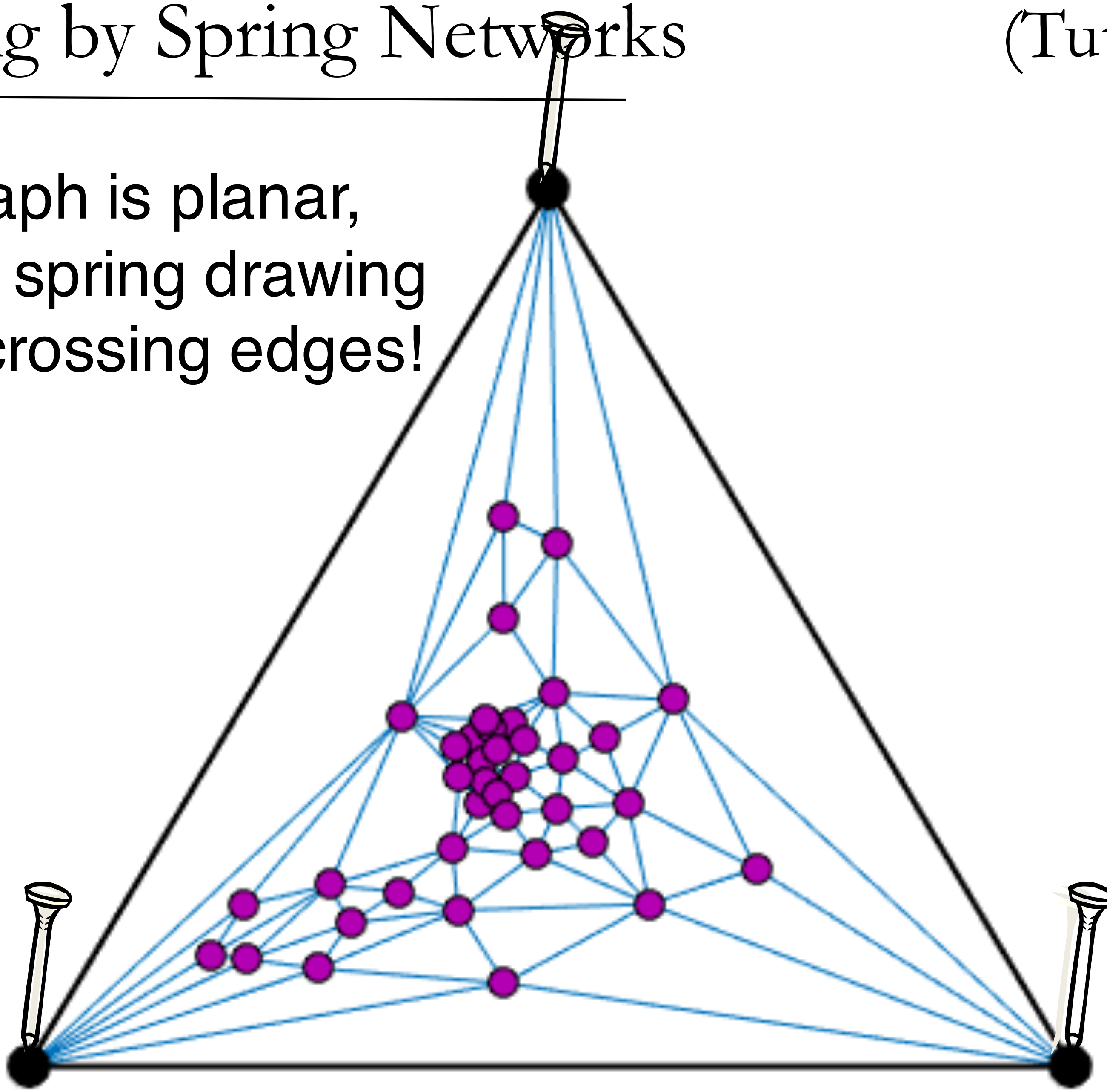


Drawing by Spring Networks (Tutte '63)

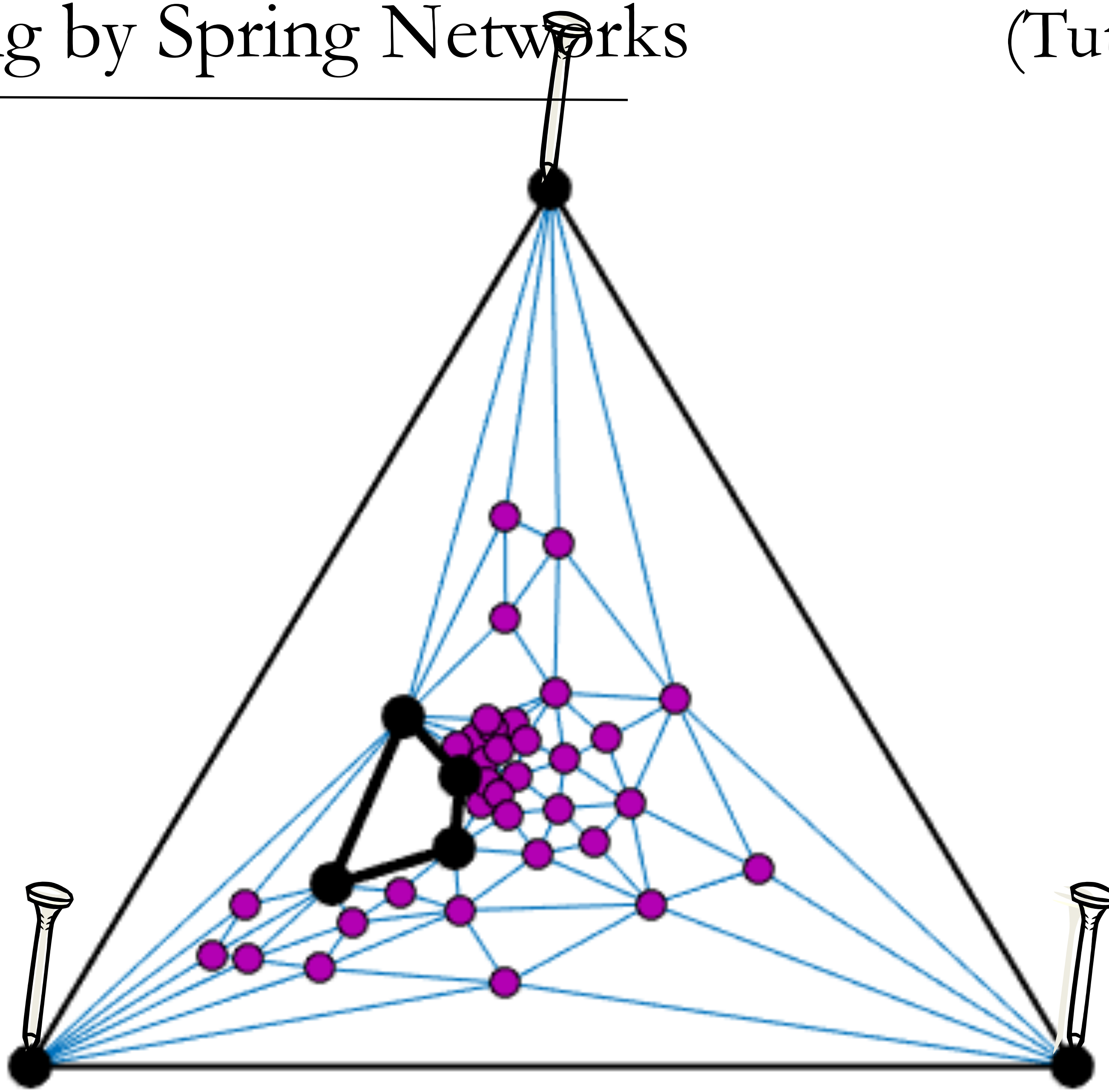


Drawing by Spring Networks (Tutte '63)

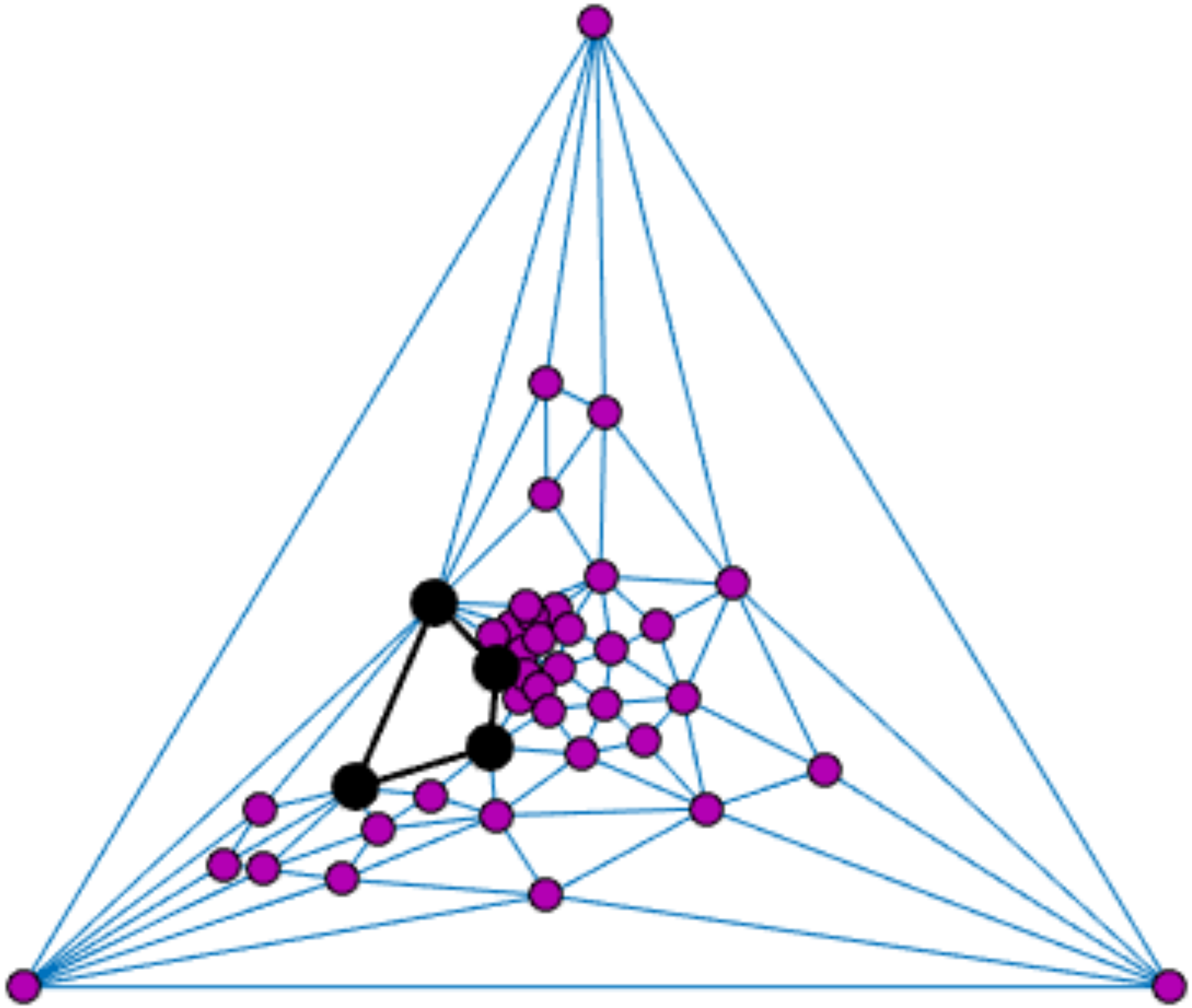
If the graph is planar, then the spring drawing has no crossing edges!



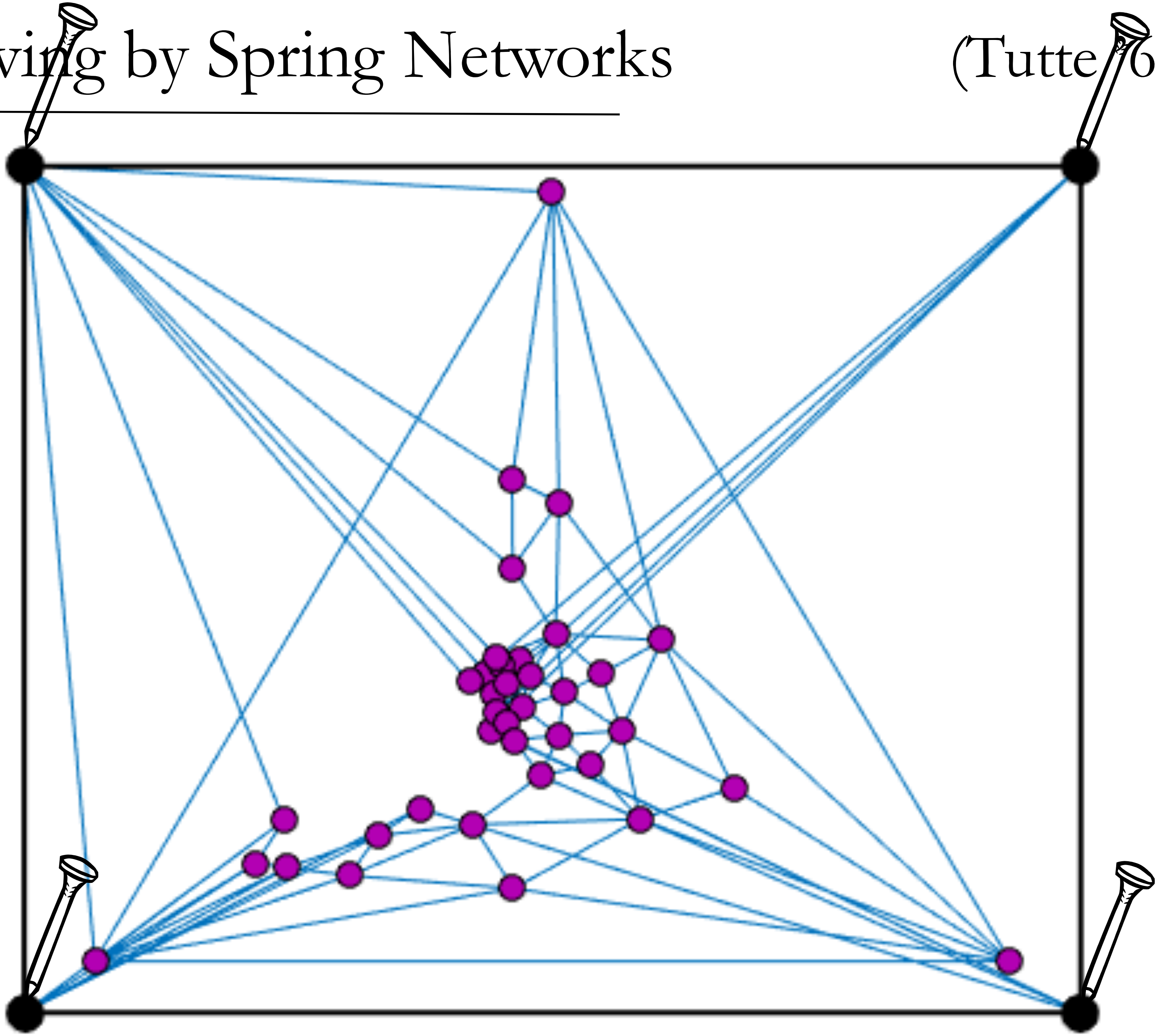
Drawing by Spring Networks (Tutte '63)



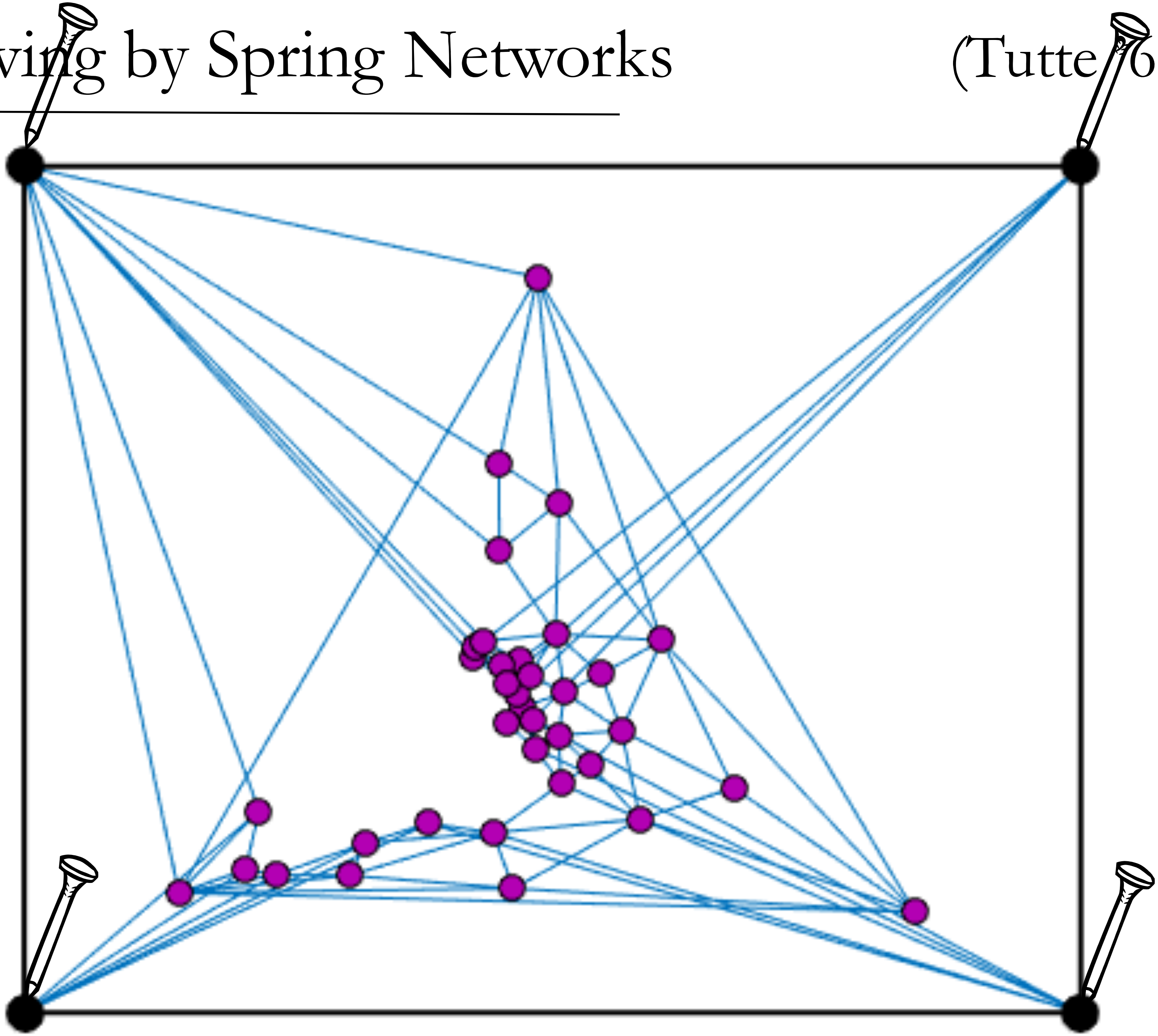
Drawing by Spring Networks (Tutte '63)



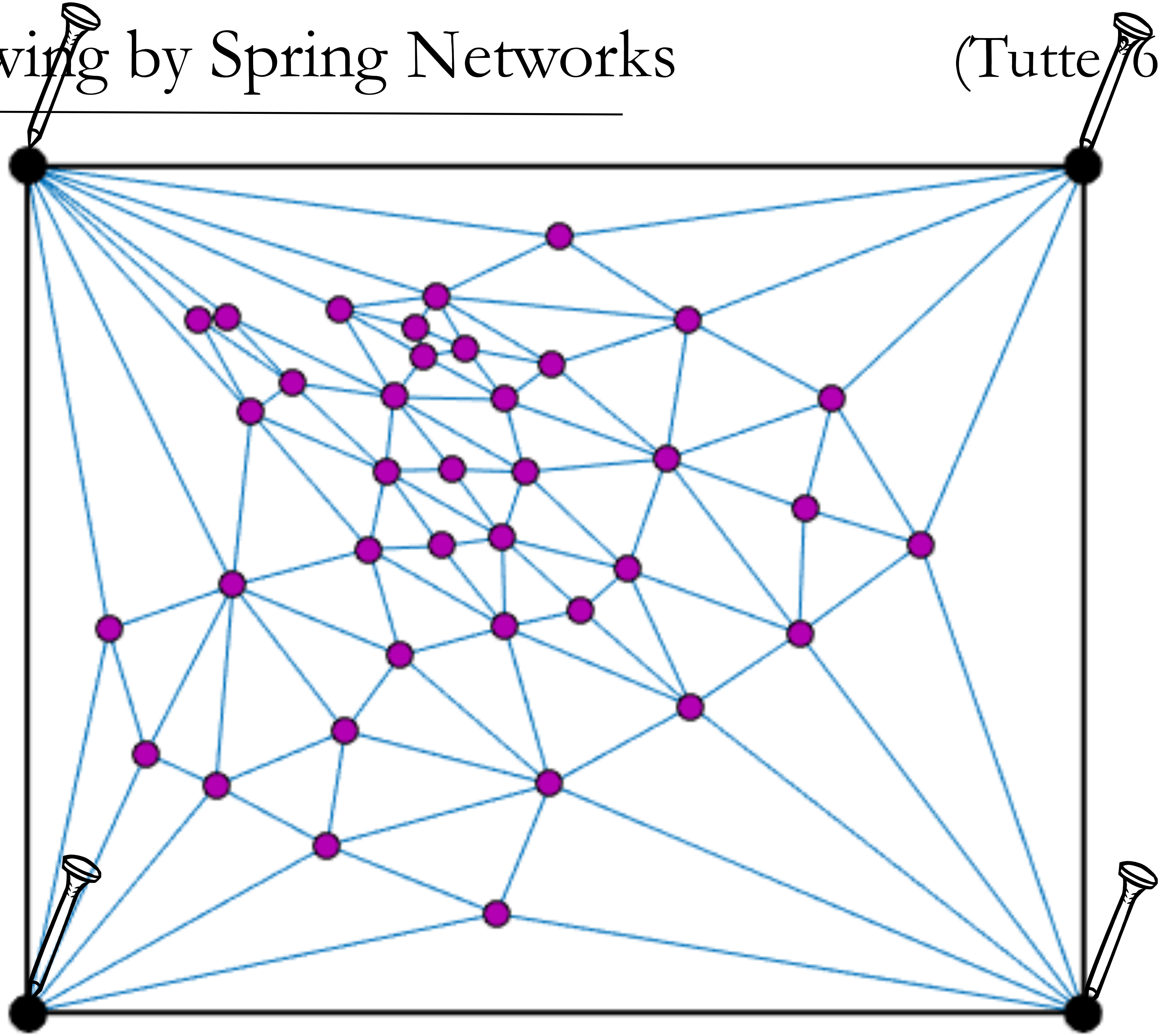
Drawing by Spring Networks (Tutte '63)



Drawing by Spring Networks (Tutte '63)

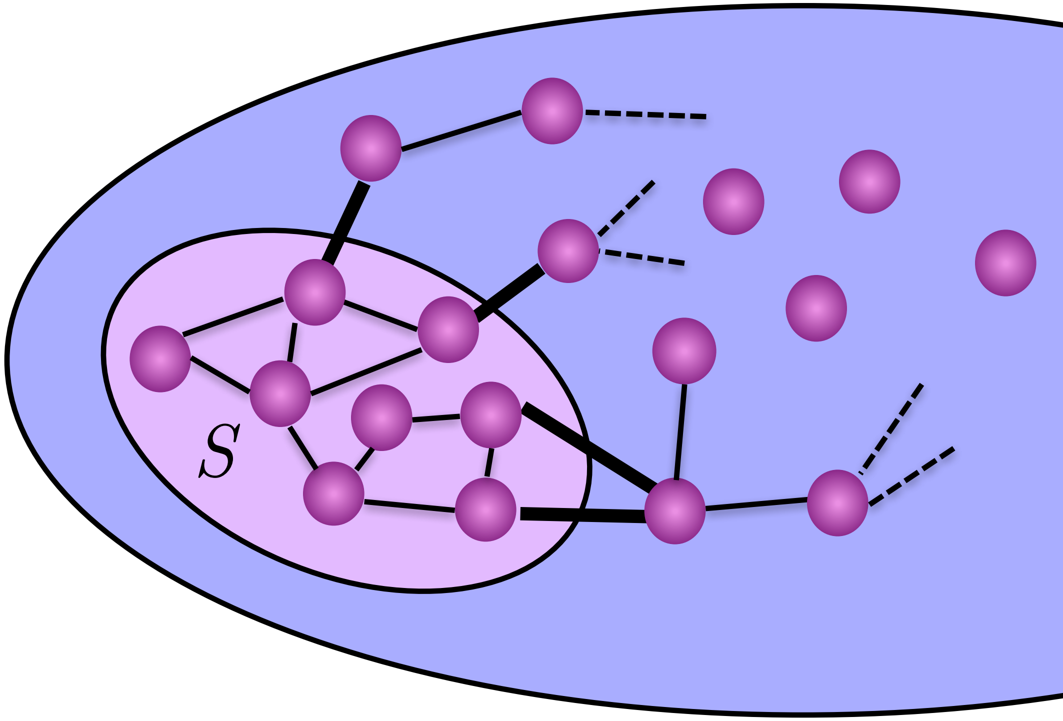


Drawing by Spring Networks (Tutte '63)



Measuring boundaries of sets

Boundary: edges leaving a set

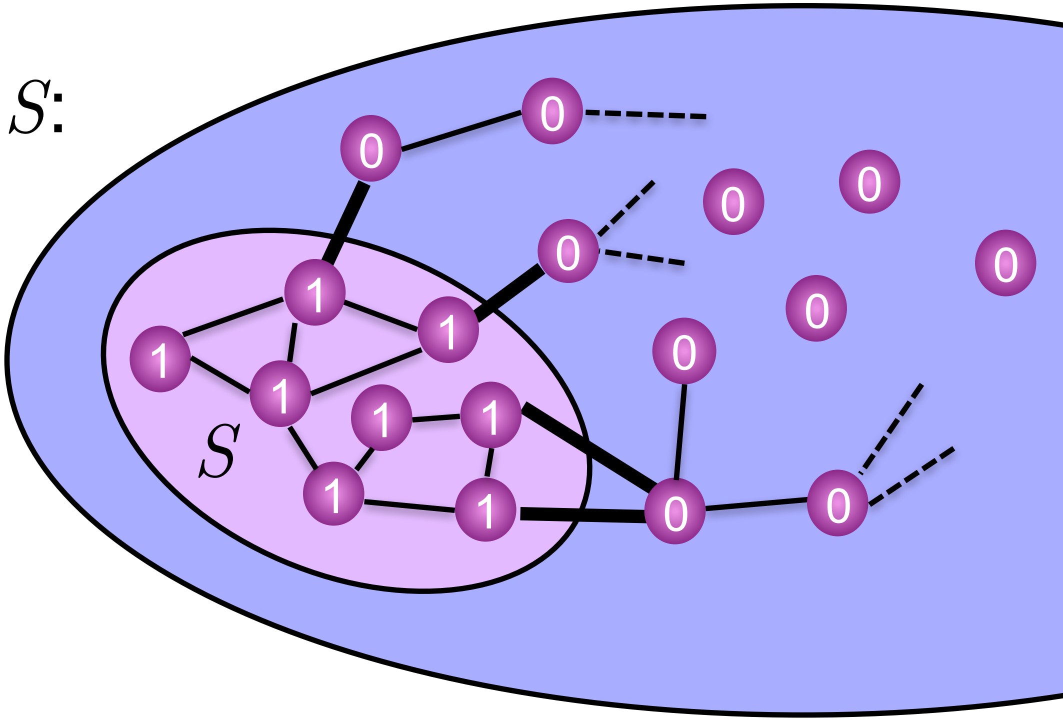


Measuring boundaries of sets

Boundary: edges leaving a set

Characteristic Vector of S:

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



Measuring boundaries of sets

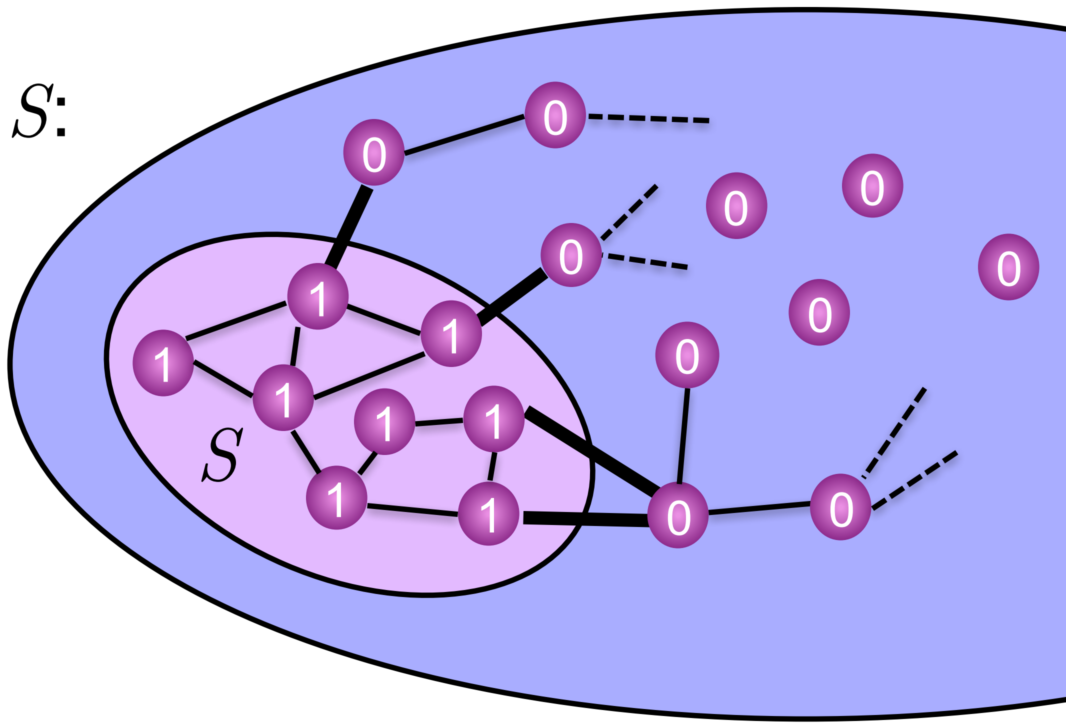
Boundary: edges leaving a set

Characteristic Vector of S:

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

$$\sum_{(i,j) \in E} (x(i) - x(j))^2$$

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



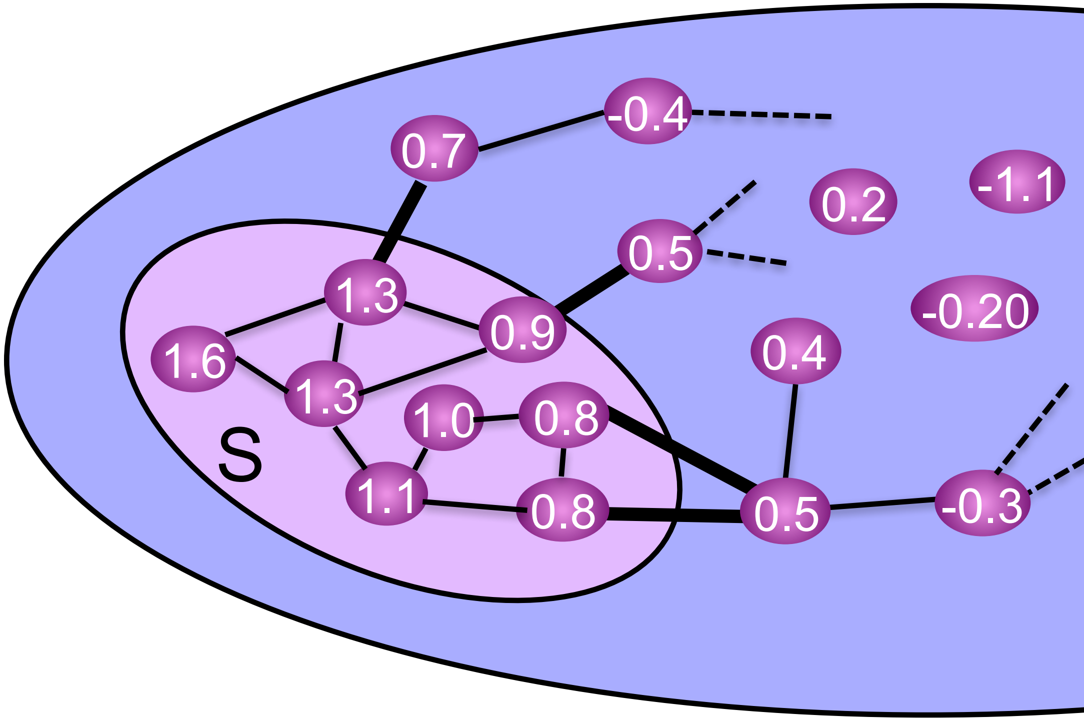
Spectral Clustering and Partitioning

Find large sets of small boundary

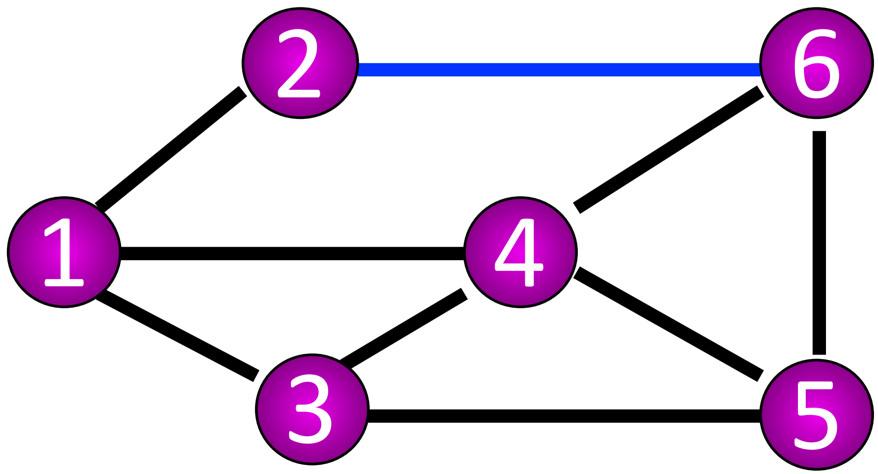
Heuristic to find
x with $x^T L_G x$ small

Compute eigenvector
$$L_G v_2 = \lambda_2 v_2$$

Consider the level sets



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

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

$$x(i) - x(j) = \begin{pmatrix} 1 & -1 \end{pmatrix} \begin{pmatrix} x(i) \\ x(j) \end{pmatrix}$$

$$(x(i) - x(j))^2 = \begin{pmatrix} x(i) \\ x(j) \end{pmatrix}^T \begin{pmatrix} 1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix}^T \begin{pmatrix} x(i) \\ x(j) \end{pmatrix}$$

$$= \begin{pmatrix} x(i) \\ x(j) \end{pmatrix}^T \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x(i) \\ x(j) \end{pmatrix}$$

Laplacian Matrices of Weighted Graphs

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

$$L_G = \sum_{(i,j)\in E} w_{i,j}(b_{i,j} b_{i,j}^T)$$ where $b_{i,j} = e_i - e_j$

Laplacian Matrices of Weighted Graphs

$$L_G = \sum_{(i,j) \in E} w_{i,j} (b_{i,j} b_{i,j}^T) \qquad \text{where } b_{i,j} = e_i - e_j$$

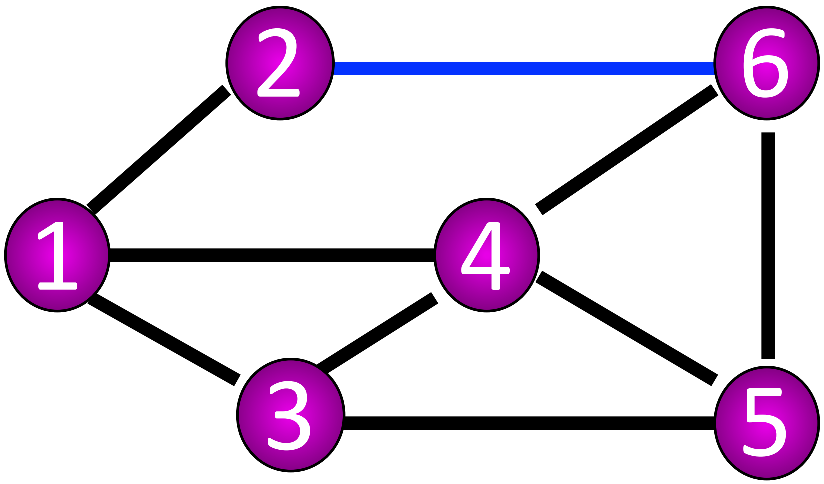
$$L_G = B^T W B$$

B is the signed edge-vertex adjacency matrix with one row for each $b_{i,j}$

W is the diagonal matrix of weights $w_{i,j}$

Laplacian Matrices of Weighted Graphs

$$L_G = \sum_{(i,j)\in E} w_{i,j}(b_{i,j}b_{i,j}^T) \qquad L_G = B^T W B$$



$$B = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

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

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:

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

Good code:

LAMG (lean algebraic multigrid) – Livne-Brandt

CMG (combinatorial multigrid) – Koutis

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 x satisfying

$$\|x - x^*\|_{L_G} \leq \epsilon \|x^*\|_{L_G}$$

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

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 x satisfying

$$\|x - x^*\|_{L_G} \leq \epsilon \|x^*\|_{L_G}$$

Allows fast computation of eigenvectors
corresponding to small eigenvalues.

Laplacians in Linear Programming

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

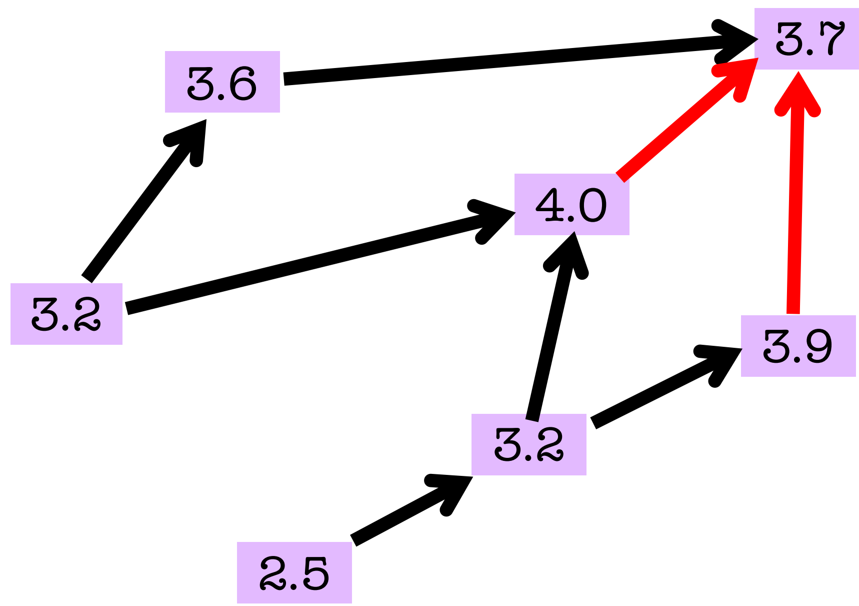
Lipschitz Learning : regularized interpolation on graphs
(Kyng, Rao, Sachdeva,S '15)

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

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

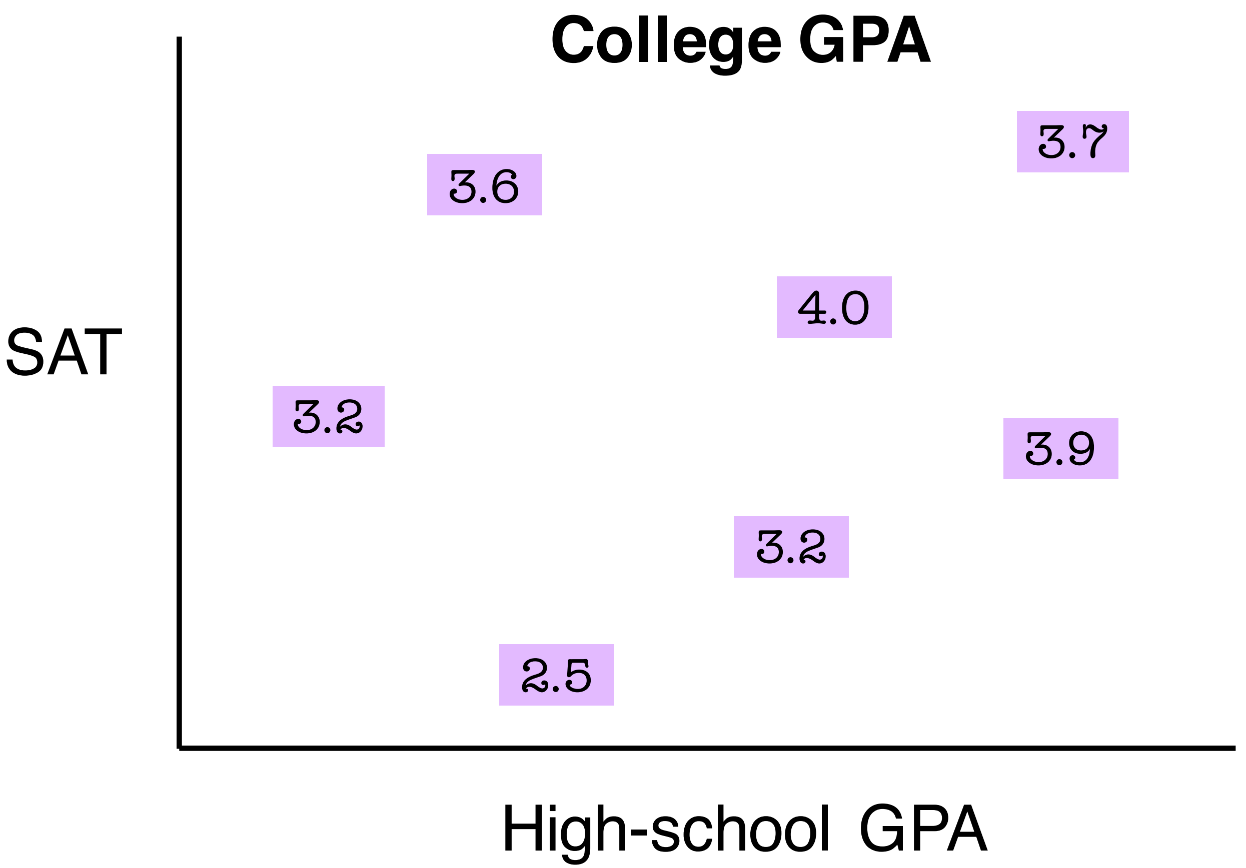
Isotonic Regression (Kyng, Rao, Sachdeva '15)

Isotonic Regression (Ayer et. al. '55)

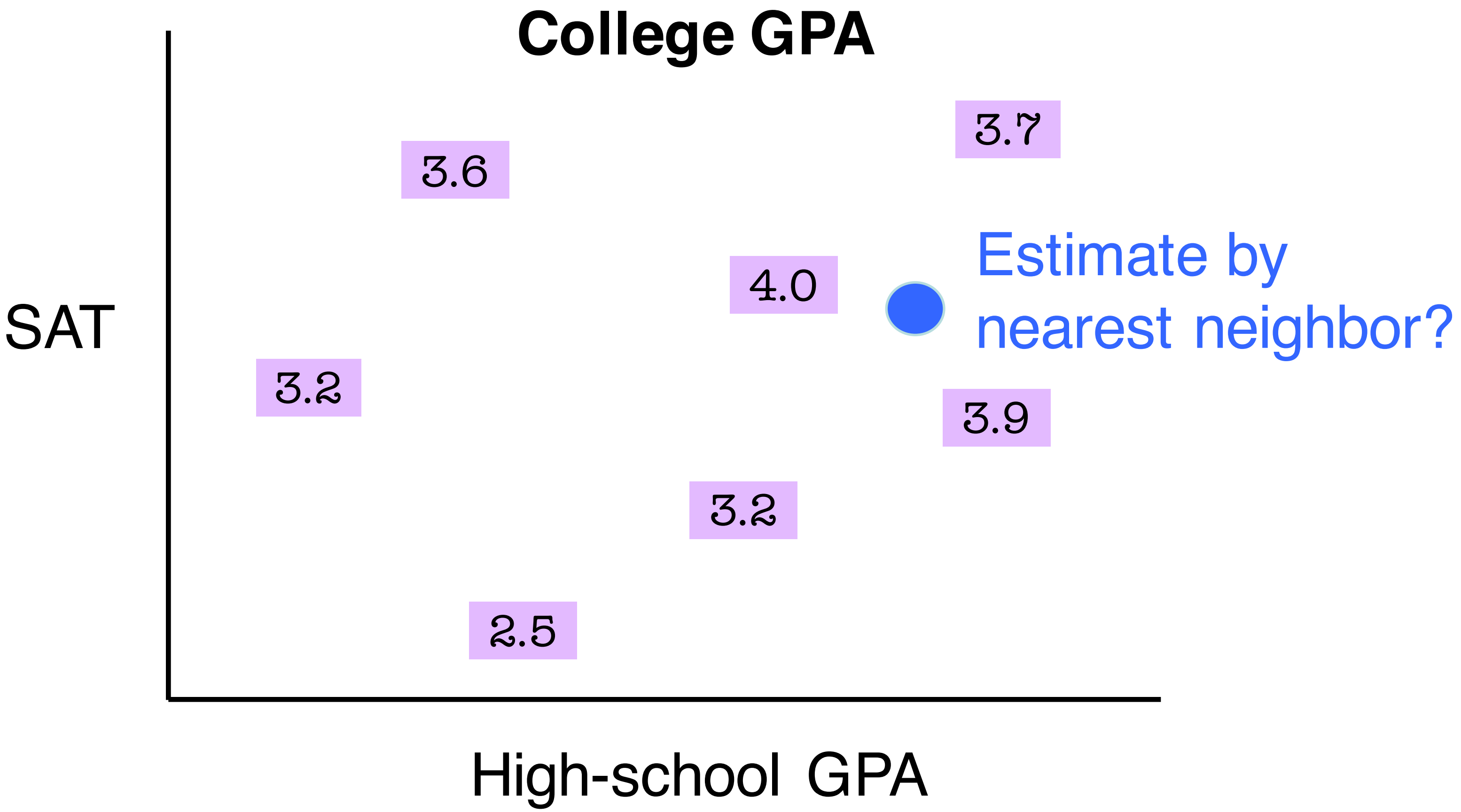


A function $x : V \to \mathbb{R}$ is isotonic with respect to a directed acyclic graph if x increases on edges.

Isotonic Regression (Ayer et. al. '55)

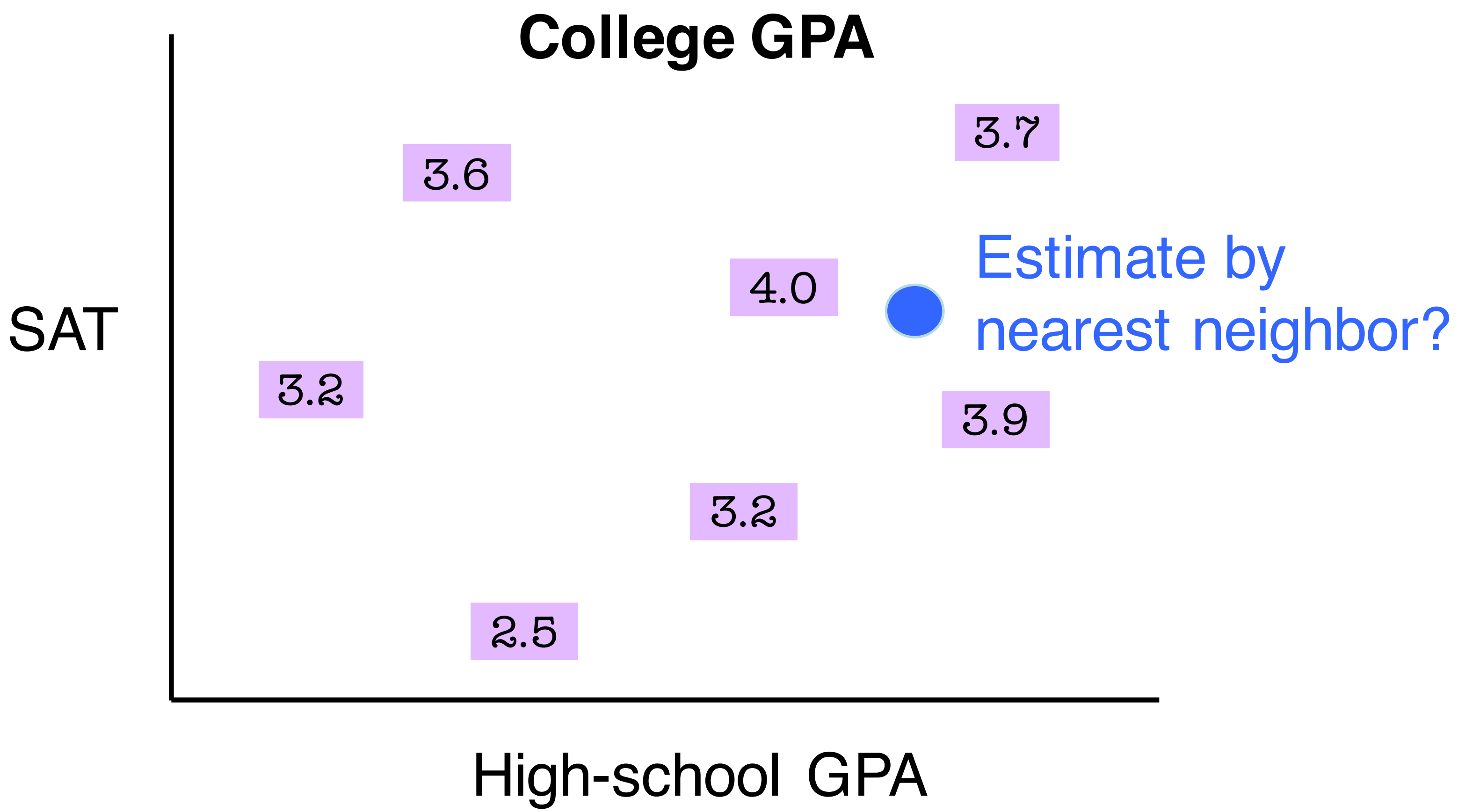


Isotonic Regression (Ayer et. al. '55)



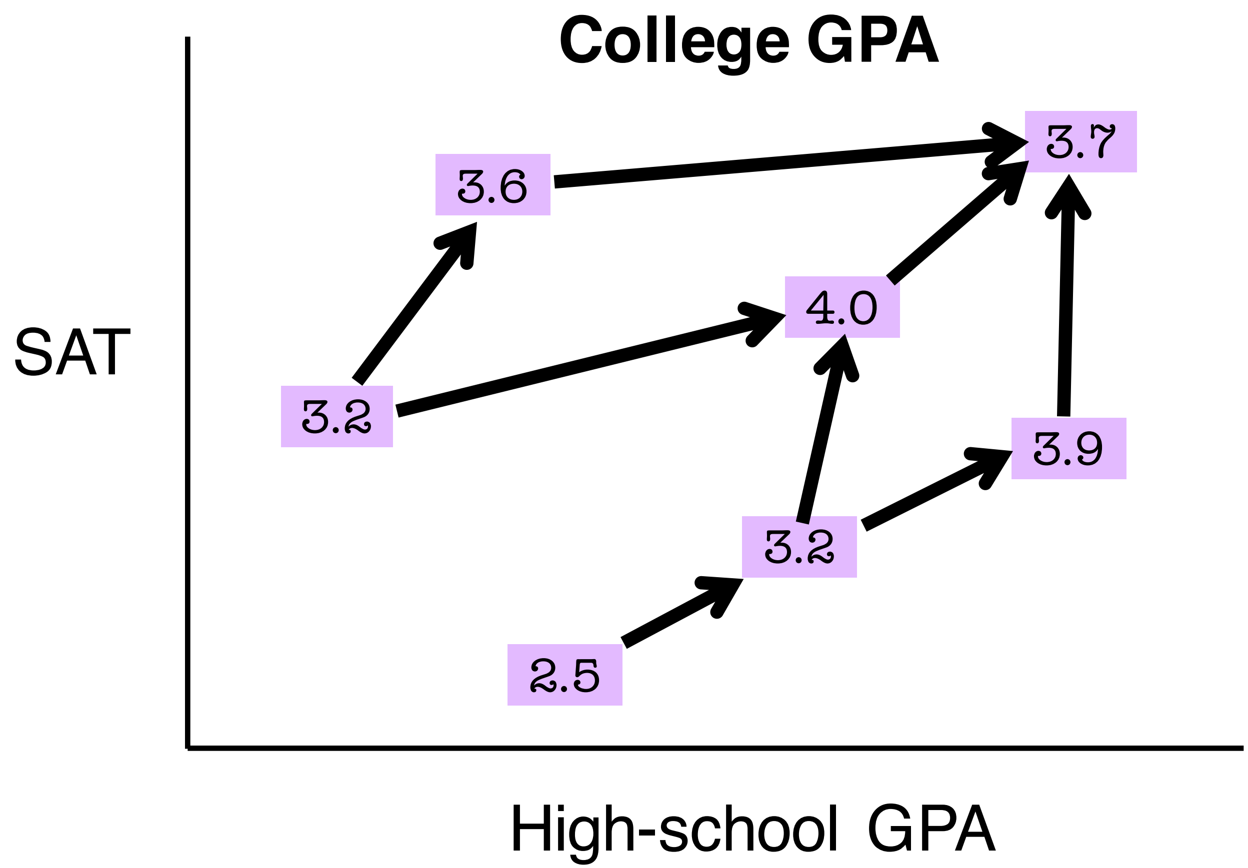
Isotonic Regression

(Ayer et. al. '55)



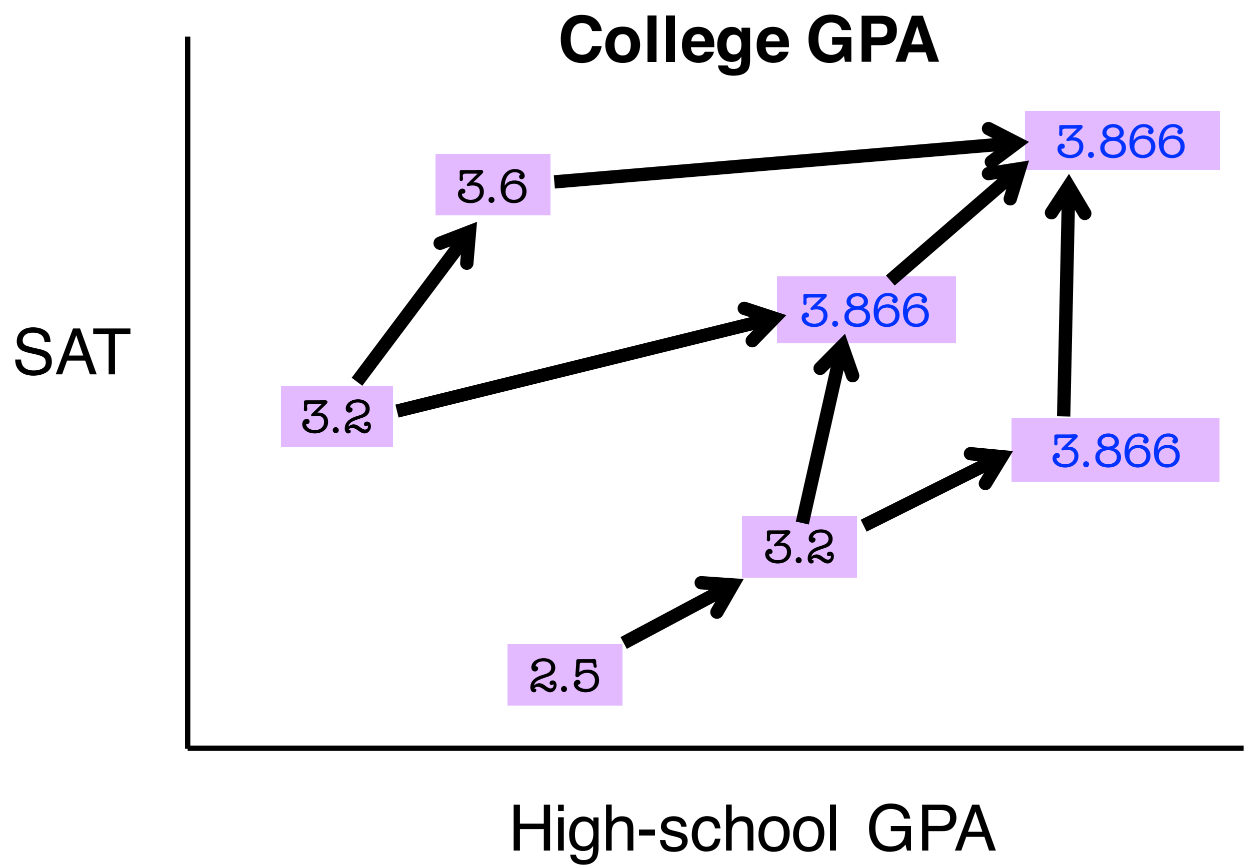
We want the estimate to be monotonically increasing

Isotonic Regression (Ayer et. al. '55)



Given $y : V \to \mathbb{R}$ find the isotonic x minimizing $\|x - y\|$

Isotonic Regression (Ayer et. al. '55)



Given $y : V \to \mathbb{R}$ find the isotonic x minimizing $\|x - y\|$

Fast IPM for Isotonic Regression

(Kyng, Rao, Sachdeva '15)

Given $y : V \to \mathbb{R}$ find the isotonic x minimizing $\|x - y\|_1$

Fast IPM for Isotonic Regression

(Kyng, Rao, Sachdeva '15)

Given $y : V \to \mathbb{R}$ find the isotonic x minimizing $\|x - y\|_1$

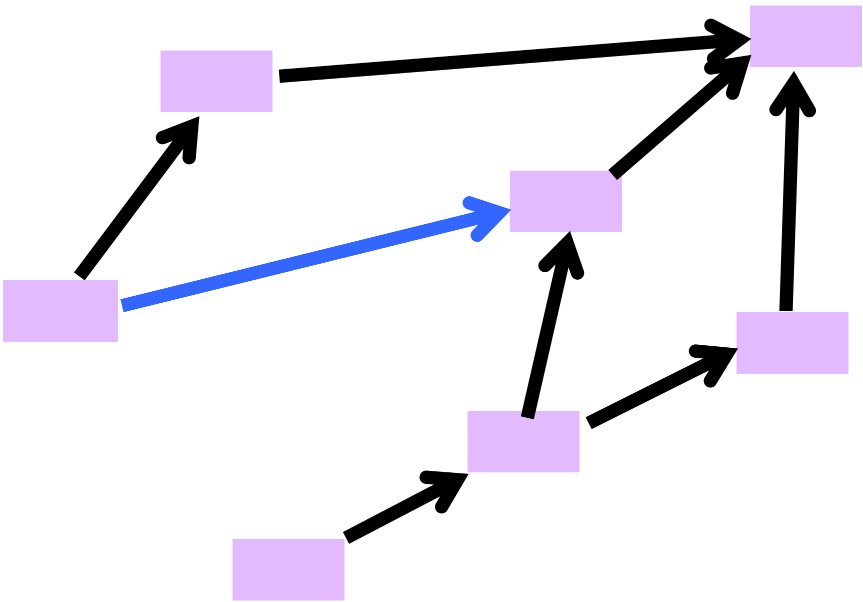
or $\|x - y\|_p$ for any $p > 1$

in time $O(m^{3/2} \log^3 m)$

Linear Program for Isotonic Regression

Signed edge-vertex incidence matrix

$$\begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$



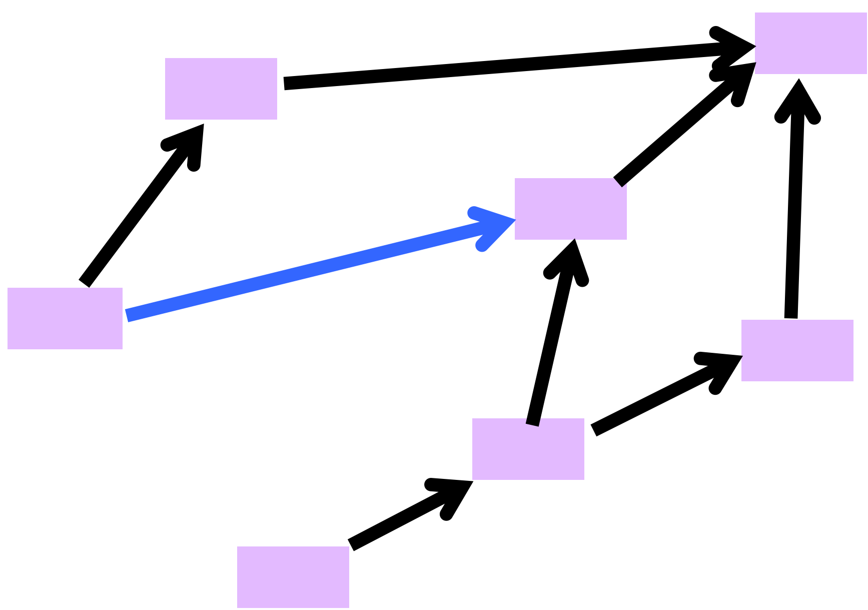
x is isotonic if $Bx \leq 0$

Linear Program for Isotonic Regression

Given y, minimize $\|x - y\|_1$

subject to $Bx \leq 0$

$$\begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$



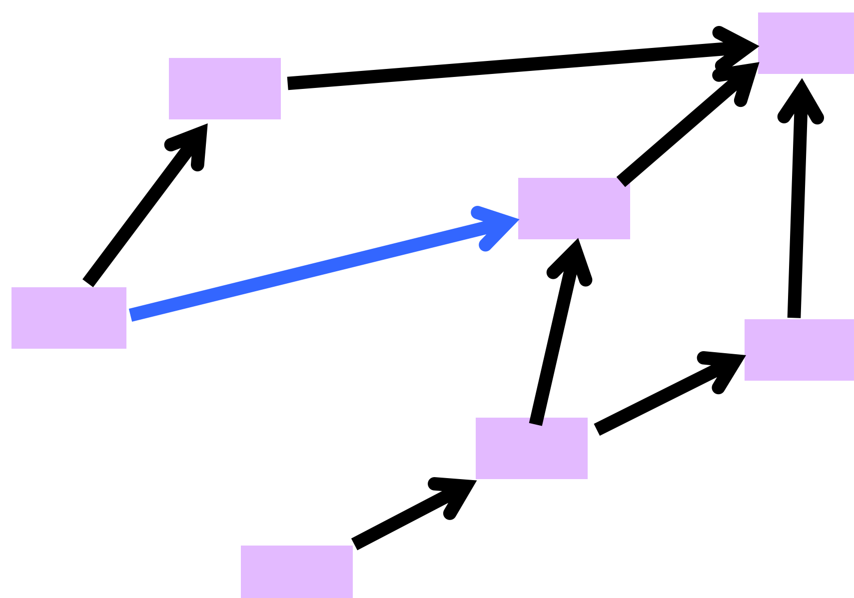
Linear Program for Isotonic Regression

Given y, minimize $\sum_i r_i$

subject to $Bx \leq 0$

$$|x_i - y_i| = r_i$$

$$\begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$



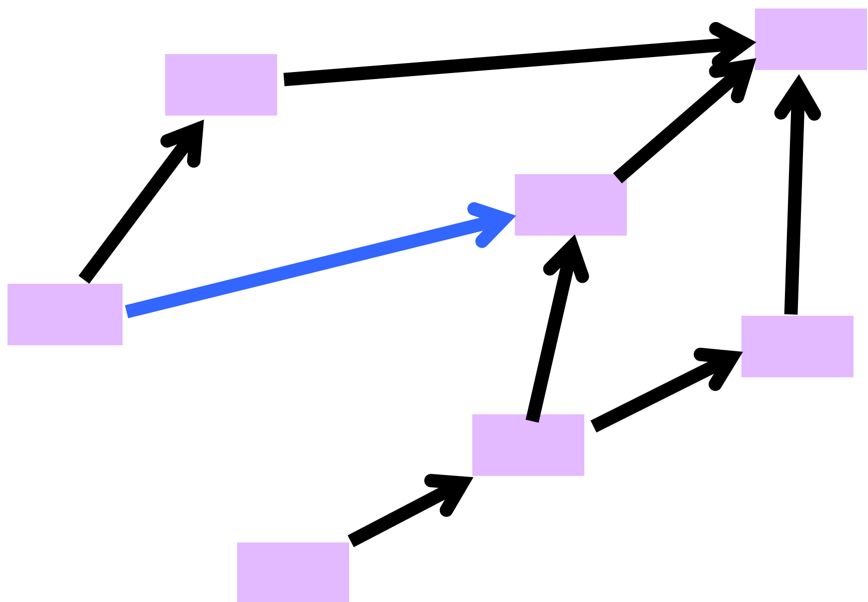
Linear Program for Isotonic Regression

Given y, minimize $\sum_i r_i$

subject to $Bx \leq 0$

$$|x_i - y_i| \leq r_i$$

$$\begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$



Linear Program for Isotonic Regression

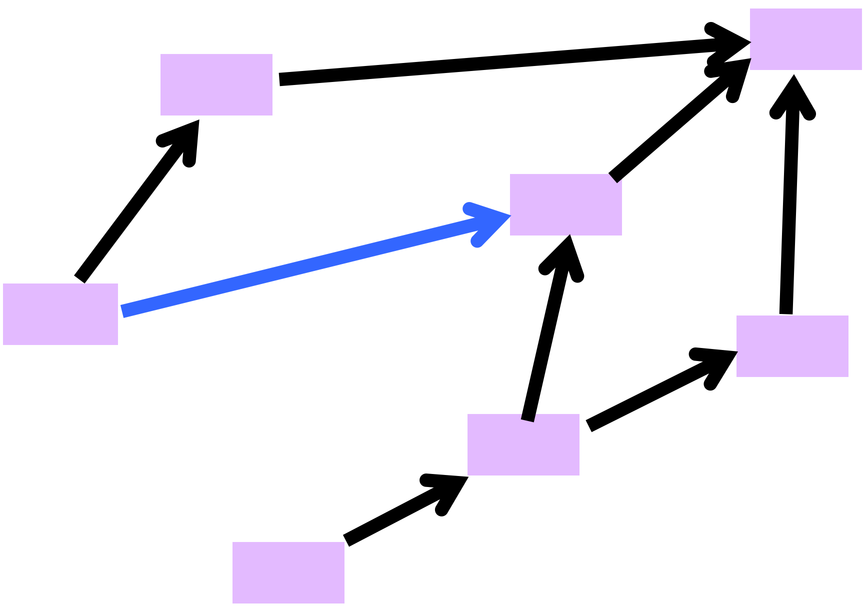
Given y, minimize $\sum_i r_i$

subject to $Bx \leq 0$

$$x_i - y_i \leq r_i$$

$$-(x_i - y_i) \leq r_i$$

$$\begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$



Linear Program for Isotonic Regression

Minimize $\sum_i r_i$

subject to
$$\begin{pmatrix} 0 & B \\ -I & I \\ -I & -I \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} \leq \begin{pmatrix} 0 \\ y \\ -y \end{pmatrix}$$

Linear Program for Isotonic Regression

Minimize $\sum_i r_i$

subject to $\begin{pmatrix} 0 & B \\ -I & I \\ -I & -I \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} \leq \begin{pmatrix} 0 \\ y \\ -y \end{pmatrix}$

IPM solves a sequence of equations of form

$$\begin{pmatrix} 0 & B \\ -I & I \\ -I & -I \end{pmatrix}^T \begin{pmatrix} S_0 & 0 & 0 \\ 0 & S_1 & 0 \\ 0 & 0 & S_2 \end{pmatrix} \begin{pmatrix} 0 & B \\ -I & I \\ -I & -I \end{pmatrix}$$

with positive diagonal matrices S_0, S_1, S_2

Linear Program for Isotonic Regression

$$\begin{pmatrix} 0 & B \\ -I & I \\ -I & -I \end{pmatrix}^T \begin{pmatrix} S_0 & 0 & 0 \\ 0 & S_1 & 0 \\ 0 & 0 & S_2 \end{pmatrix} \begin{pmatrix} 0 & B \\ -I & I \\ -I & -I \end{pmatrix}$$

$$= \begin{pmatrix} S_1 + S_2 & S_2 - S_1 \\ S_2 - S_1 & \underbrace{B^T S_0 B}_{\text{Laplacian!}} + S_1 + S_2 \end{pmatrix}$$

S_0, S_1, S_2 are positive diagonal

Linear Program for Isotonic Regression

$$\begin{pmatrix} 0 & B \\ -I & I \\ -I & -I \end{pmatrix}^T \begin{pmatrix} S_0 & 0 & 0 \\ 0 & S_1 & 0 \\ 0 & 0 & S_2 \end{pmatrix} \begin{pmatrix} 0 & B \\ -I & I \\ -I & -I \end{pmatrix}$$

$$= \begin{pmatrix} S_1 + S_2 & S_2 - S_1 \\ S_2 - S_1 & \underbrace{B^T S_0 B}_{\text{Laplacian!}} + S_1 + S_2 \end{pmatrix}$$

S_0, S_1, S_2 are positive diagonal

Kyng, Rao, Sachdeva '15:
Reduce to solving Laplacians to constant accuracy

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

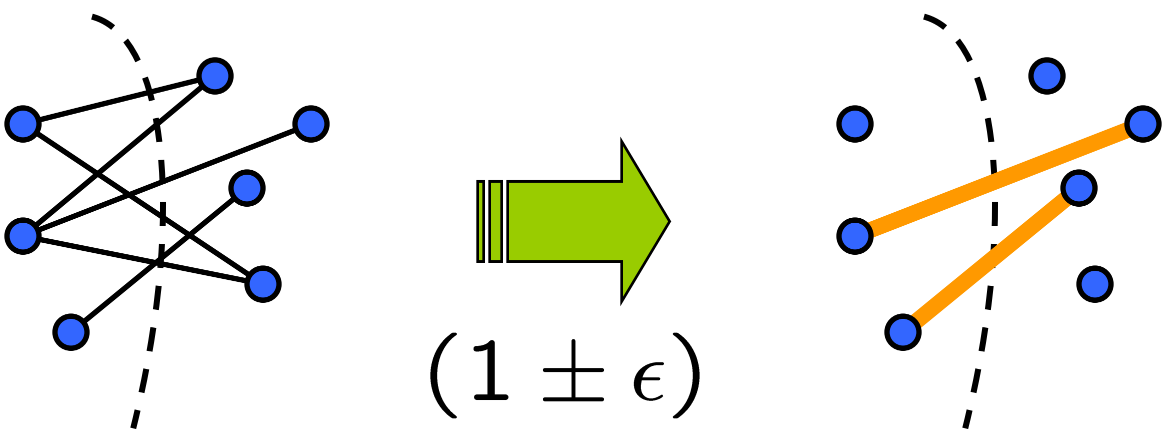
$$L_H \approx_\epsilon L_G$$

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

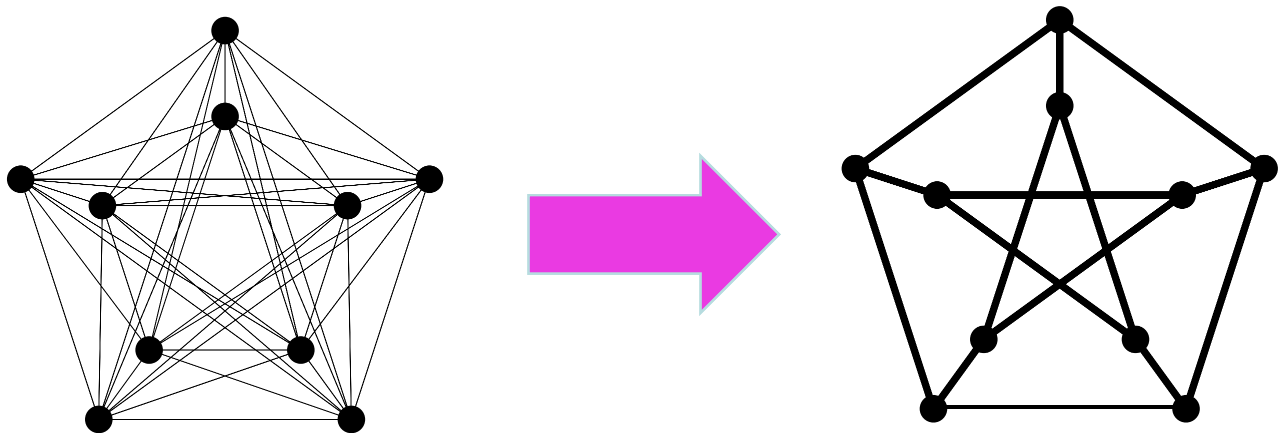
Solutions to linear equations are similiar

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

Spectral Sparsification

(Batson, S, Srivastava '09)

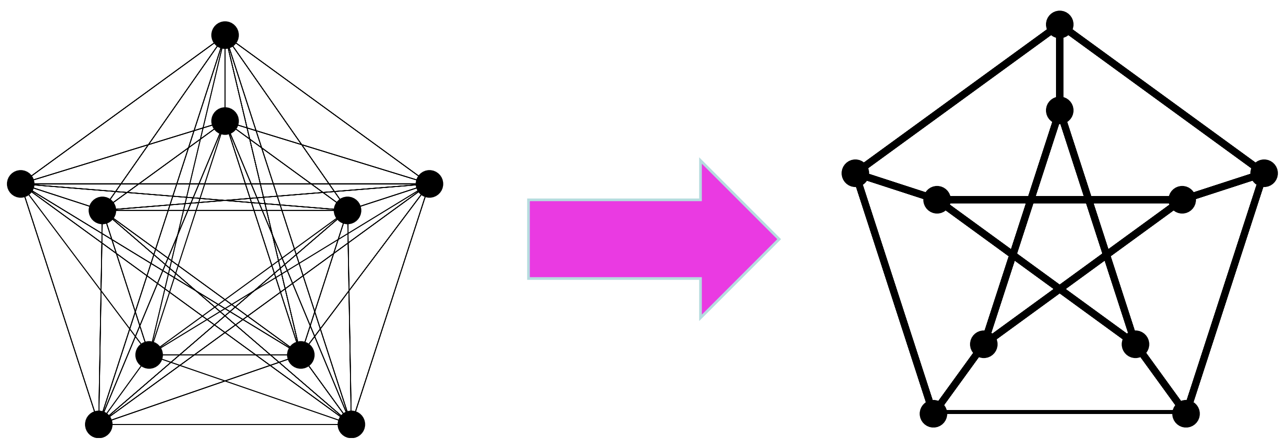
Every graph G has an ϵ-approximation H with $n(2+\epsilon)^2/\epsilon^2$ edges



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 spring constant, only need $O(n \log n / \epsilon^2)$ samples

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

(Lee & Sun '15)

Can find an ϵ-approximation with $O(n/\epsilon^2)$ edges in time $O(n^{1+c})$ for every $c > 0$

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

$$L_G \approx U^T U$$

Gaussian Elimination

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

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

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

2. Subtract it

Gaussian Elimination

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

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

2. Subtract it

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

3. Repeat

Gaussian Elimination

2. Subtract it

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

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

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

Gaussian Elimination

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

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

2. Subtract it

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

Gaussian Elimination

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

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

Gaussian Elimination

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

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

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

Gaussian Elimination

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

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

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

Gaussian Elimination

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

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

Computation time proportional to the sum of the squares of the number of nonzeros in these vectors

Gaussian Elimination of Laplacians

If this is a Laplacian, then so is this

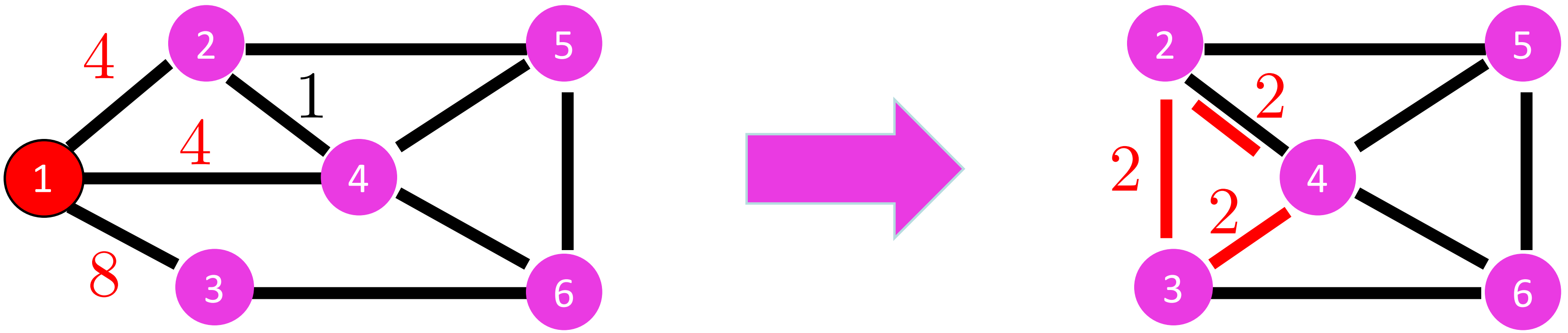
$$\begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 5 & 0 & -1 \\ -8 & 0 & 14 & 0 \\ -4 & -1 & 0 & 7 \end{pmatrix} - \begin{pmatrix} 4 \\ -1 \\ -2 \\ -1 \end{pmatrix} \begin{pmatrix} 4 \\ -1 \\ -2 \\ -1 \end{pmatrix}^T = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 4 & -2 & -2 \\ 0 & -2 & 10 & -2 \\ 0 & -2 & -2 & 6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

Gaussian Elimination of Laplacians

If this is a Laplacian, then so is this

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

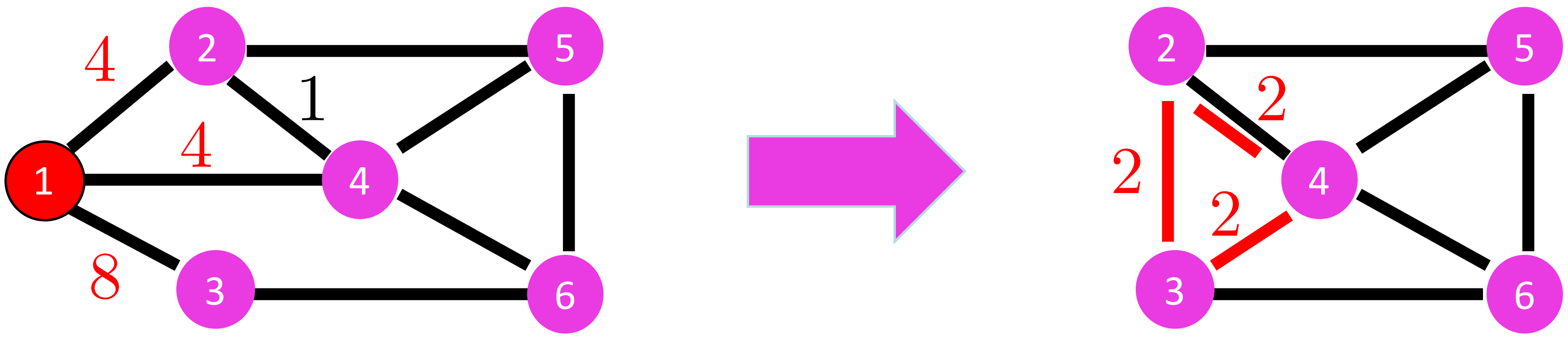
When eliminate a node, add a clique on its neighbors



Approximate Gaussian Elimination

(Kyng & Sachdeva '16)

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



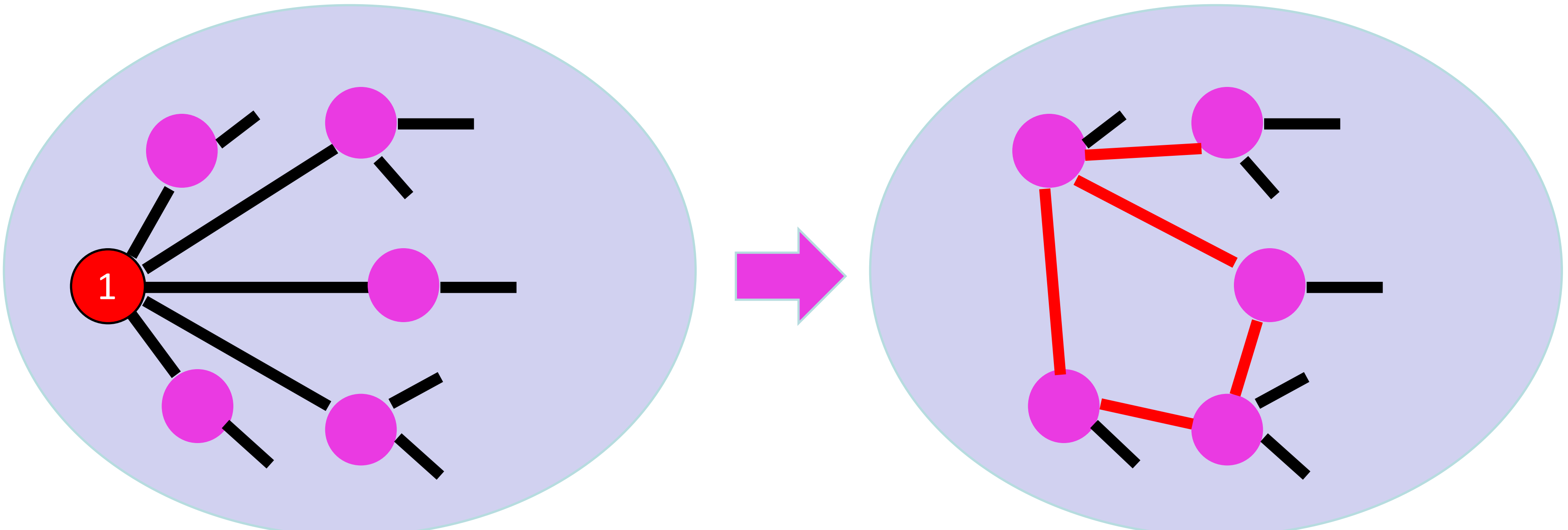
2. Sparsify that clique, without ever constructing it

Approximate Gaussian Elimination

(Kyng & Sachdeva '16)

1. 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)

0. Initialize by randomly permuting vertices, and making $O(\log^2 n)$ copies of every edge

1. 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

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

Approximate Gaussian Elimination

(Kyng & Sachdeva ‘16)

0. Initialize by randomly permuting vertices, and making $O(\log^2 n)$ copies of every edge

1. 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

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

Can be improved by sacrificing some simplicity

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)

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

Laplacians.jl

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"

$Lx = b$, by Nisheeth Vishnoi