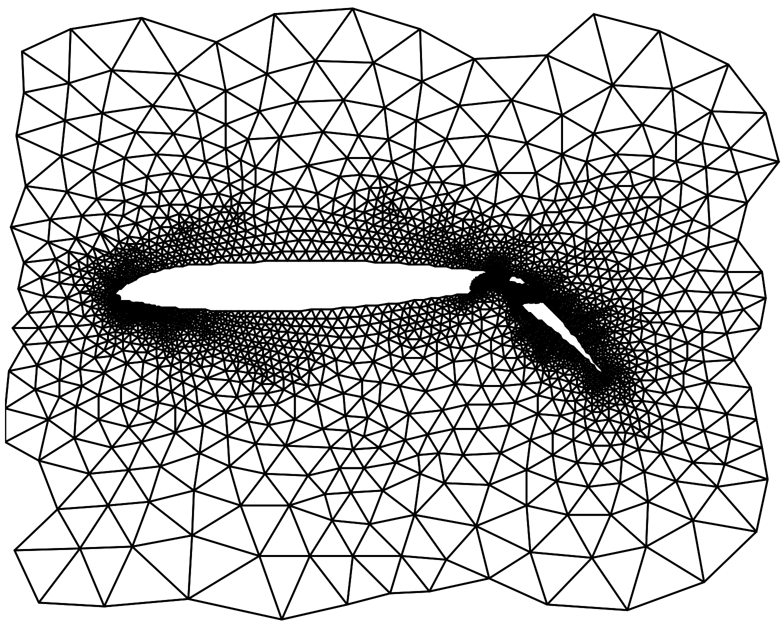
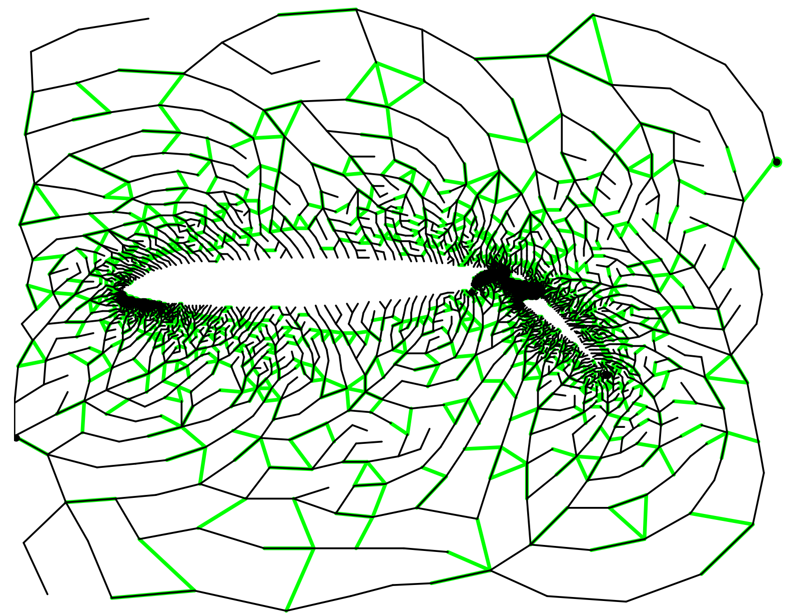
Solving Systems of Linear Equations in Graph Laplacians





Daniel A. Spielman
Yale University

Toronto, Sep. 29, 2011

Outline

Linear Systems in Laplacian Matrices

Classic ways to solve them

Approximating Graphs by Trees

Sparse Approximations of Graphs

Fast Solution of Linear Equations

Solving Linear Equations $Ax = b$, Quickly

Solve in time $O(m \log^c m)$
where m = number of non-zeros entries of A

times $\log(1/\epsilon)$ for ϵ-approximate solution.

Special case: A is the Laplacian Matrix of a Graph

Solving Linear Equations $Ax = b$, Quickly

Solve in time $O(m \log^c m)$
where m = number of non-zeros entries of A

times $\log(1/\epsilon)$ for ϵ-approximate solution.

$$\left\|x - A^{-1}b\right\|_A \leq \epsilon \left\|A^{-1}b\right\|_A$$

Special case: A is the Laplacian Matrix of a Graph

Solving Linear Equations $Ax = b$, Quickly

Solve in time $O(m \log^c m)$

where m = number of non-zeros entries of A

times $\log(1/\epsilon)$ for ϵ-approximate solution.

$$\left\|x - A^{-1}b\right\|_A \leq \epsilon \left\|A^{-1}b\right\|_A$$

where $\|x\|_A \stackrel{\text{def}}{=} \sqrt{x^T A x}$

Special case: A is the Laplacian Matrix of a Graph

Laplacian Quadratic Form of $G = (V,E)$

For $\boldsymbol{x} : V \to \mathbb{R}$

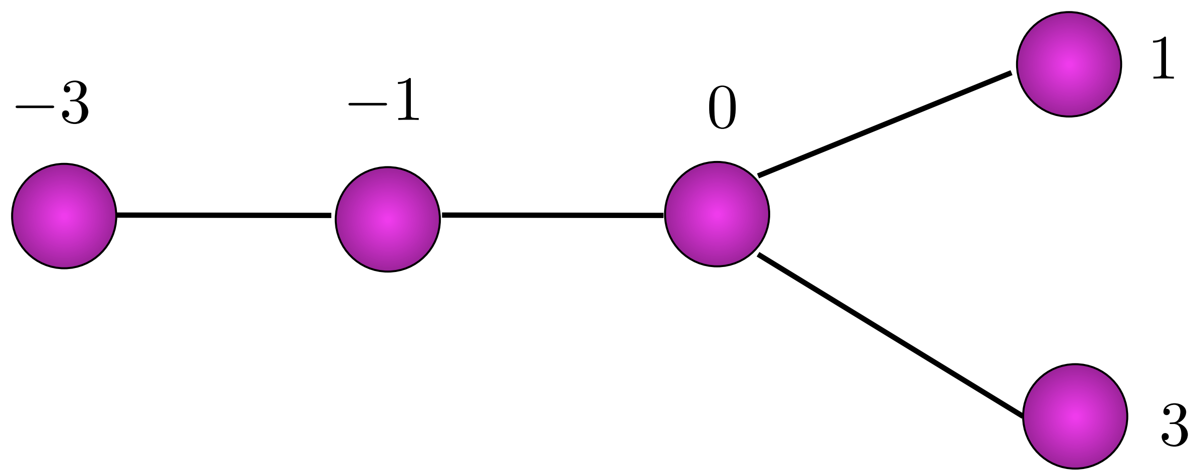
$$\boxed{\boldsymbol{x}^T L_G \boldsymbol{x} = \sum_{(u,v) \in E} \left(\boldsymbol{x}(u) - \boldsymbol{x}(v) \right)^2}$$

Laplacian Quadratic Form of $G = (V,E)$

For $\boldsymbol{x} : V \to \mathbb{R}$

$$\boxed{\boldsymbol{x}^T L_G \boldsymbol{x} = \sum_{(u,v)\in E} \left(\boldsymbol{x}(u) - \boldsymbol{x}(v)\right)^2}$$

$\boldsymbol{x}$:

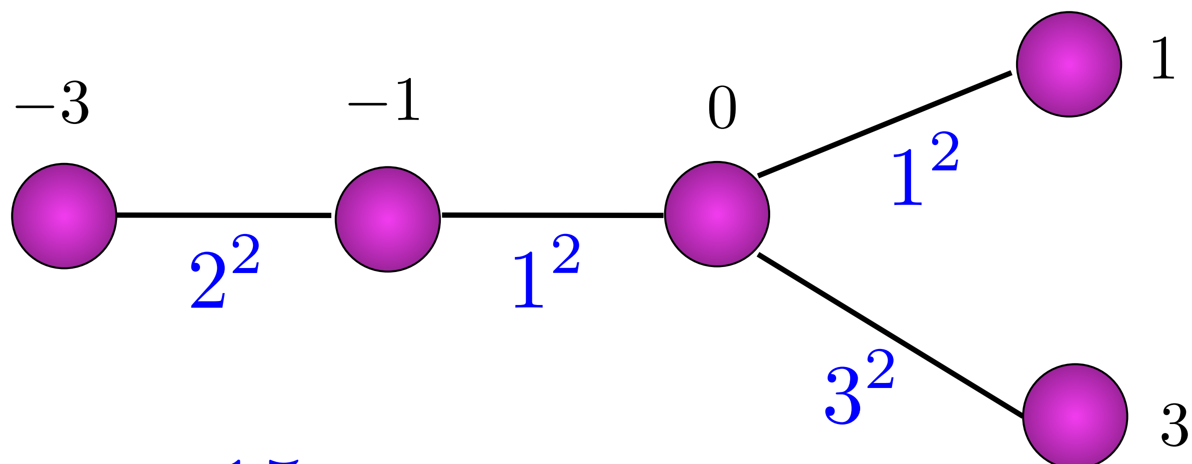


Laplacian Quadratic Form of $G = (V,E)$

For $\boldsymbol{x} : V \to \mathbb{R}$

$$\boxed{\boldsymbol{x}^T L_G \boldsymbol{x} = \sum_{(u,v)\in E} \left(\boldsymbol{x}(u) - \boldsymbol{x}(v)\right)^2}$$

$\boldsymbol{x}$:



$$\boldsymbol{x}^T L_G \boldsymbol{x} = 15$$

Laplacian Quadratic Form for Weighted Graphs

$$G = (V, E, w)$$

$w : E \to \mathbb{R}^+$ assigns a positive weight to every edge

$$\boxed{\boldsymbol{x}^T L_G \boldsymbol{x} = \sum_{(u,v) \in E} w_{(u,v)} \left(\boldsymbol{x}(u) - \boldsymbol{x}(v)\right)^2}$$

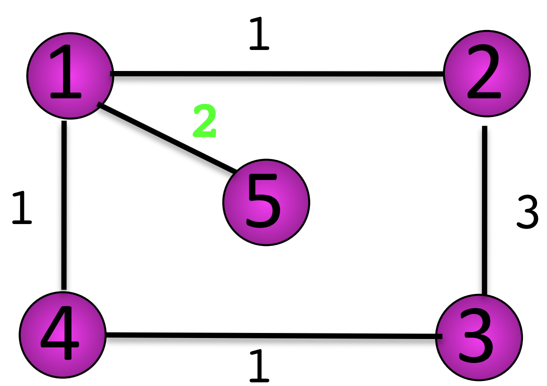
Matrix L_G is positive semi-definite
nullspace spanned by const vector, if connected

Laplacian Matrix of a Weighted Graph

$$L_G(u,v) = \begin{cases} -w(u,v) & \text{if } (u,v) \in E \\ d(u) & \text{if } u = v \\ 0 & \text{otherwise} \end{cases}$$

$$d(u) = \sum_{(v,u)\in E} w(u,v)$$

the weighted degree of u



4	−1	0	−1	−2
−1	4	−3	0	0
0	−3	4	−1	0
−1	0	−1	2	0
−2	0	0	0	2

is a diagonally dominant matrix

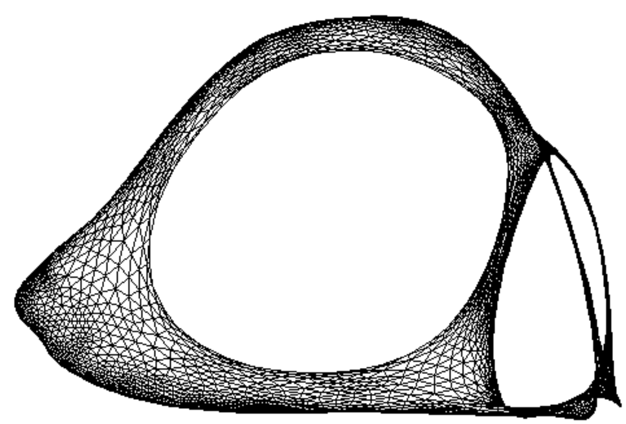
A few applications

Computing effective resistances.

Solving Elliptic PDEs.

Solving Maximum Flow by Interior Point Methods

Computing Eigenvectors and Eigenvalues of
Laplacians of graphs.



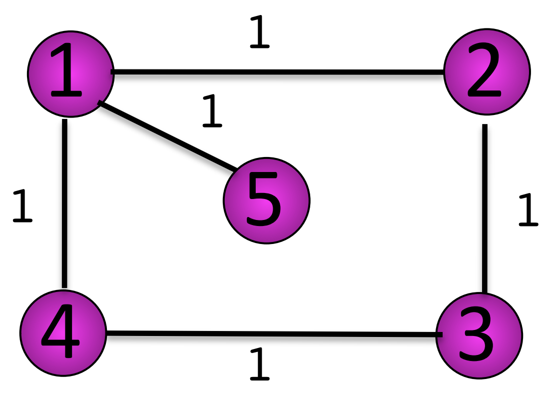
Solving Laplacian Linear Equations Quickly

Fast when graph is simple,
by elimination.

Fast approximation when graph is complicated*,
by Conjugate Gradient

* = random graph or high expansion

Cholesky Factorization of Laplacians

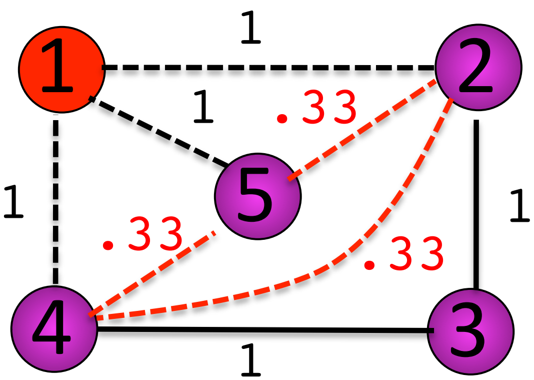


$$\begin{matrix} 3 & -1 & 0 & -1 & -1 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & 0 & -1 & 2 & 0 \\ -1 & 0 & 0 & 0 & 1 \end{matrix}$$

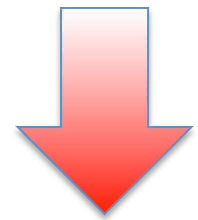
When eliminate a vertex,
connect its neighbors.

Also known as Y-Δ

Cholesky Factorization of Laplacians



3	-1	0	-1	-1
-1	2	-1	0	0
0	-1	2	-1	0
-1	0	-1	2	0
-1	0	0	0	1

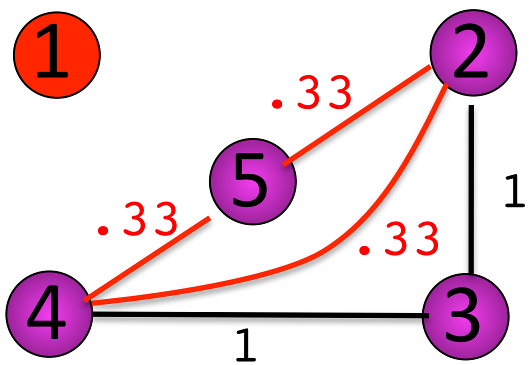


When eliminate a vertex, connect its neighbors.

3	0	0	0	0
0	1.67	-1.00	-0.33	-0.33
0	-1.00	2.00	-1.00	0
0	-0.33	-1.00	1.67	-0.33
0	-0.33	0	-0.33	0.67

Also known as Y-Δ

Cholesky Factorization of Laplacians

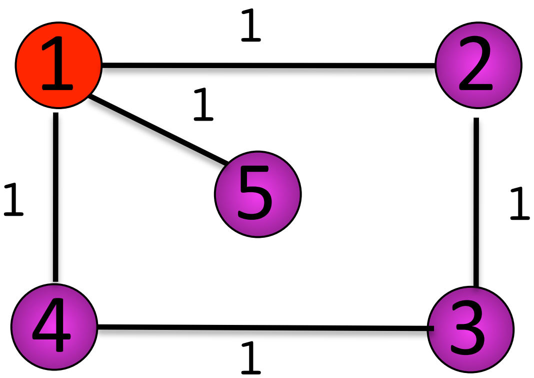


3	-1	0	-1	-1
-1	2	-1	0	0
0	-1	2	-1	0
-1	0	-1	2	0
-1	0	0	0	1

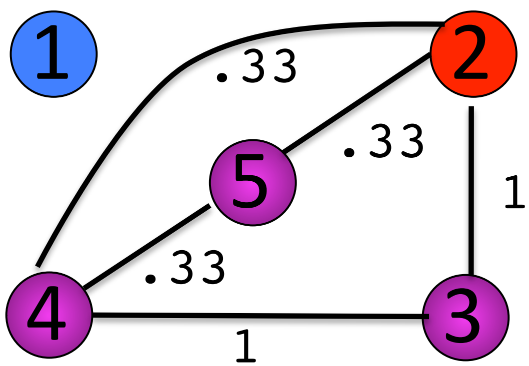
3	0	0	0	0
0	1.67	-1.00	-0.33	-0.33
0	-1.00	2.00	-1.00	0
0	-0.33	-1.00	1.67	-0.33
0	-0.33	0	-0.33	0.67

When eliminate a vertex, connect its neighbors.

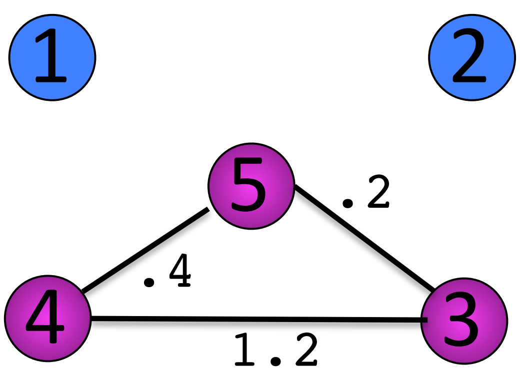
Also known as Y-Δ



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

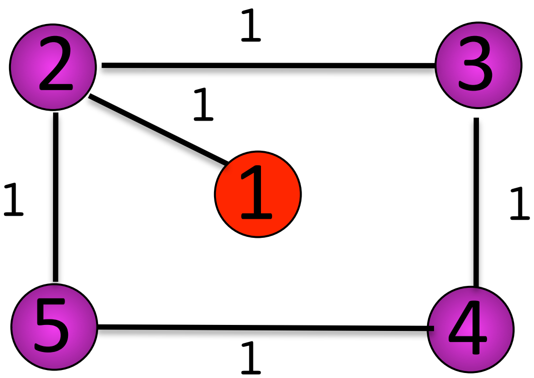


$$\begin{matrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 1.67 & -1.00 & -0.33 & -0.33 \\ 0 & -1.00 & 2.00 & -1.00 & 0 \\ 0 & -0.33 & -1.00 & 1.67 & -0.33 \\ 0 & -0.33 & 0 & -0.33 & 0.67 \end{matrix}$$

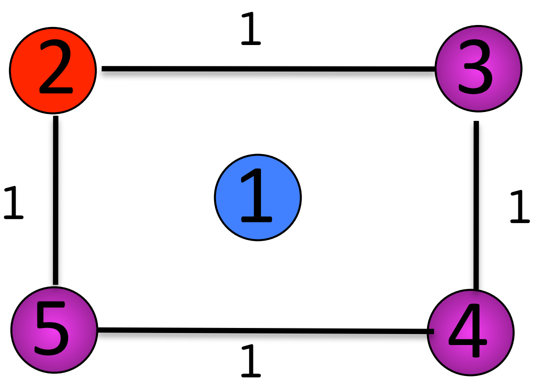


$$\begin{matrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 1.67 & 0 & 0 & 0 \\ 0 & 0 & 1.4 & -1.2 & -0.2 \\ 0 & 0 & -1.2 & 1.6 & -0.4 \\ 0 & 0 & -0.2 & -0.4 & 0.6 \end{matrix}$$

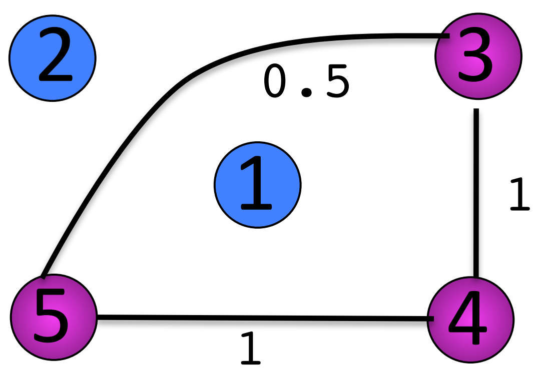
The order matters



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



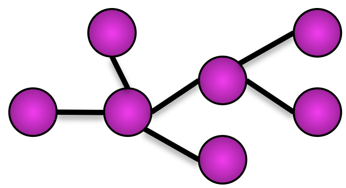
$$\begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & -1 & 0 & -1 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & 0 & -1 & 2 \end{matrix}$$



$$\begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1.5 & -1 & -0.5 \\ 0 & 0 & -1.0 & 2 & -1.0 \\ 0 & 0 & -0.5 & -1 & 1.5 \end{matrix}$$

Complexity of Cholesky Factorization

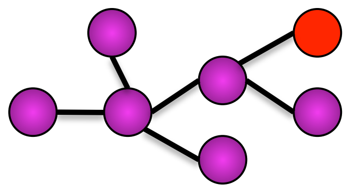
#ops ~ Σ_v (degree of v when eliminate)^2

Tree  #ops ~ $O(|V|)$

Complexity of Cholesky Factorization

#ops ~ Σ_v (degree of v when eliminate)^2

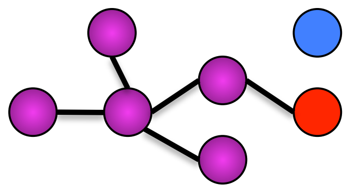
Tree



#ops ~ O(|V|)

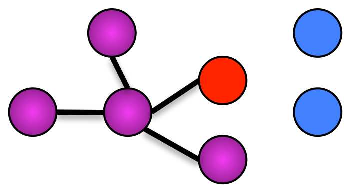
Complexity of Cholesky Factorization

#ops ~ Σ_v (degree of v when eliminate)^2

Tree  #ops ~ O(|V|)

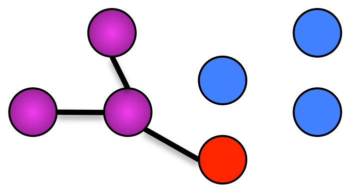
Complexity of Cholesky Factorization

#ops ~ Σ_v (degree of v when eliminate)^2

Tree  #ops ~ O(|V|)

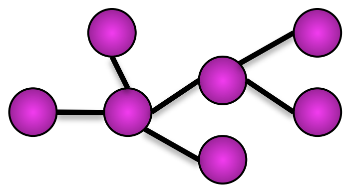
Complexity of Cholesky Factorization

#ops ~ Σ_v (degree of v when eliminate)^2

Tree  #ops ~ O(|V|)

Complexity of Cholesky Factorization

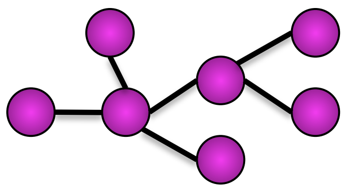
#ops ~ Σ_v (degree of v when eliminate)^2

Tree  #ops ~ O(|V|)

Complexity of Cholesky Factorization

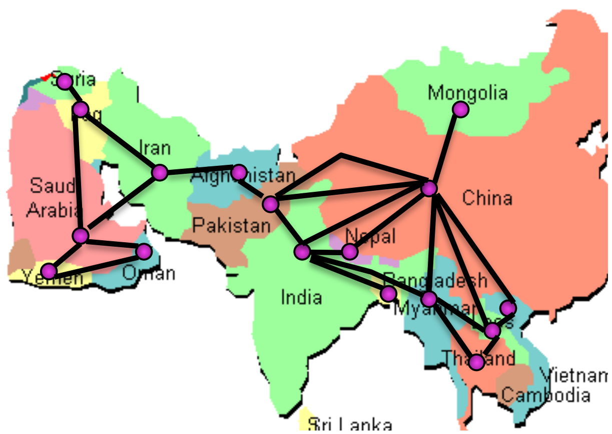
#ops ~ Σ_v (degree of v when eliminate)^2

Tree



#ops ~ $O(|V|)$

Planar



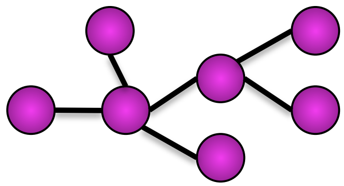
#ops ~ $O(|V|^{3/2})$

Lipton-Rose-Tarjan '79

Complexity of Cholesky Factorization

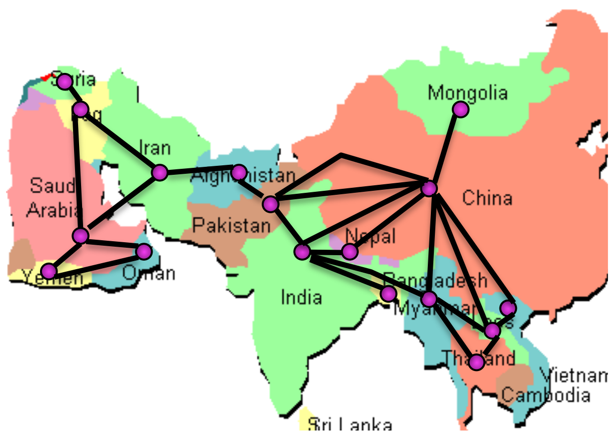
#ops ~ Σ_v (degree of v when eliminate)^2

Tree



#ops ~ $O(|V|)$

Planar



#ops ~ $O(|V|^{3/2})$
Lipton-Rose-Tarjan '79

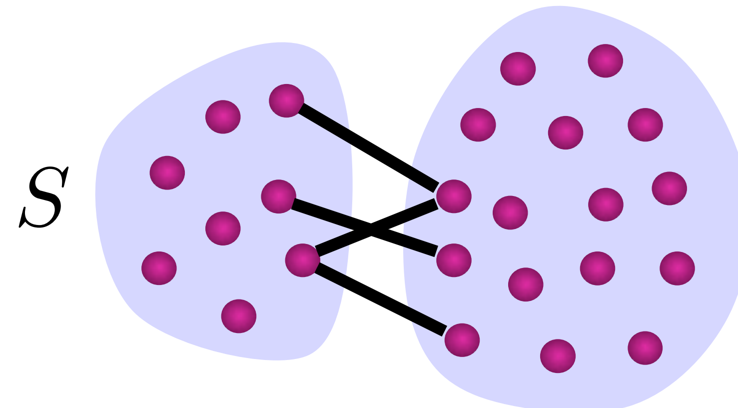
Expander

like random,
but $O(|V|)$ edges

#ops $\gtrsim \Omega(|V|^3)$
Lipton-Rose-Tarjan '79

Expansion and Cholesky Factorization

For $S \subset V$

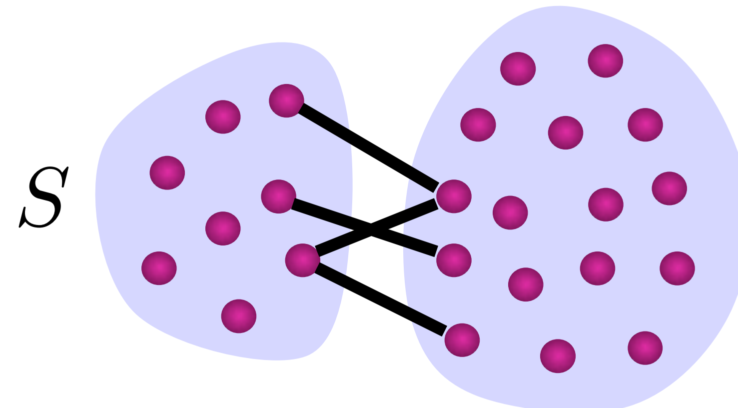


$$\Phi(S) = \frac{|\mathrm{bdry}(S)|}{\min\left(|S|, |V - S|\right)}$$

$$\boxed{\Phi_G = \min_{S \subset V} \Phi(S)}$$

Expansion and Cholesky Factorization

For $S \subset V$



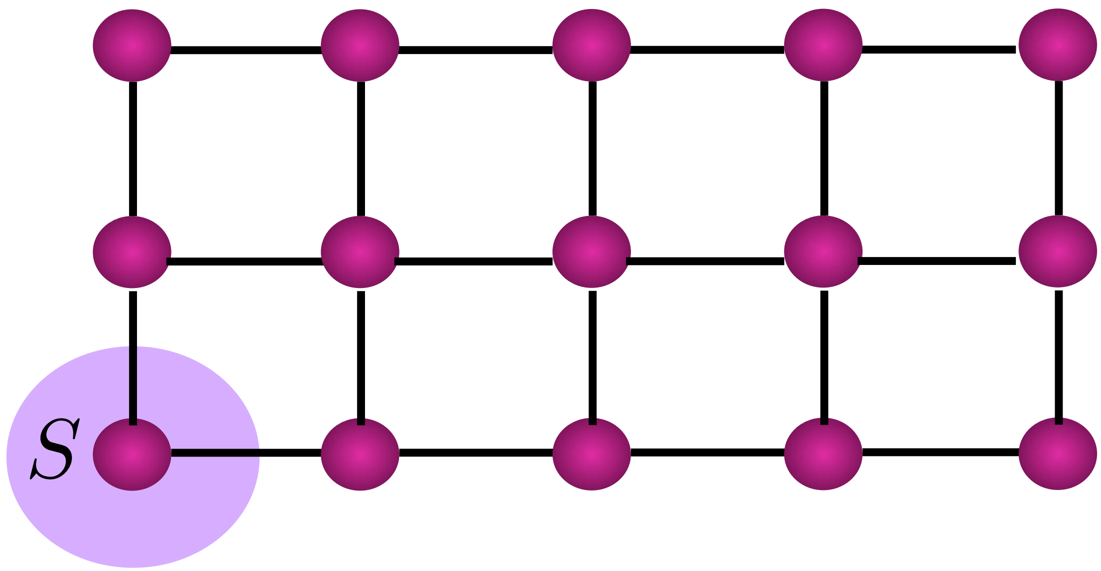
$$\Phi(S) = \frac{|\mathrm{bdry}(S)|}{\min\left(|S|, |V - S|\right)}$$

$$\boxed{\Phi_G = \min_{S \subset V} \Phi(S)}$$

Cholesky slow when expansion high
Cholesky fast when low for G and all subgraphs

Expansion

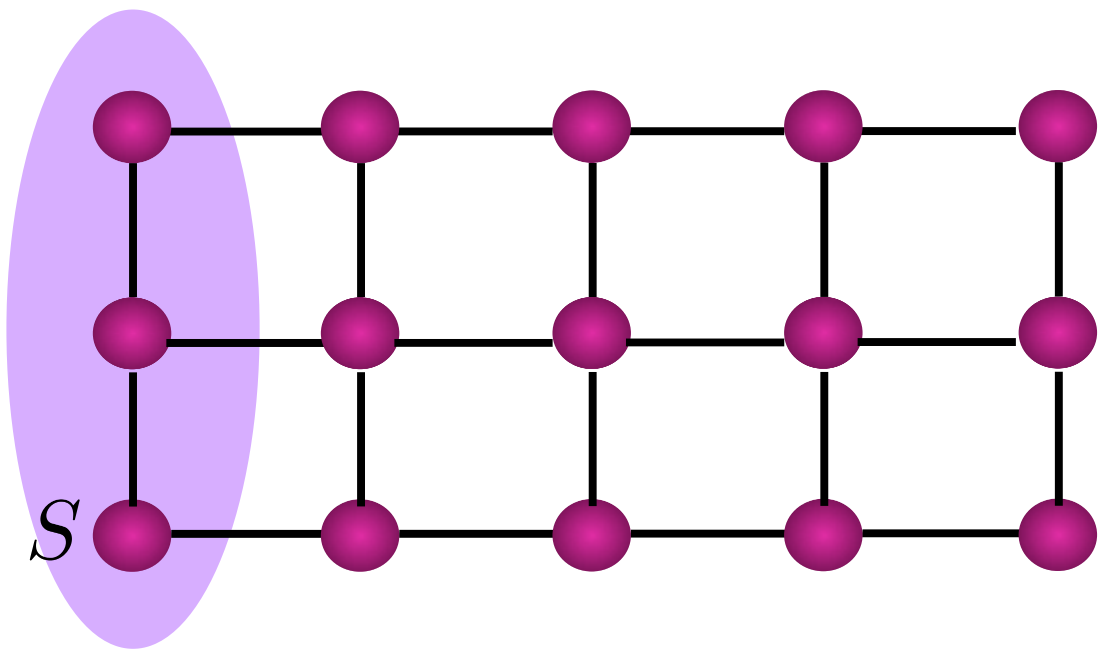
$$\Phi(S) = \frac{|\mathrm{bdry}(S)|}{\min\left(|S|, |V - S|\right)}$$



$$\Phi(S) = 2$$

Expansion

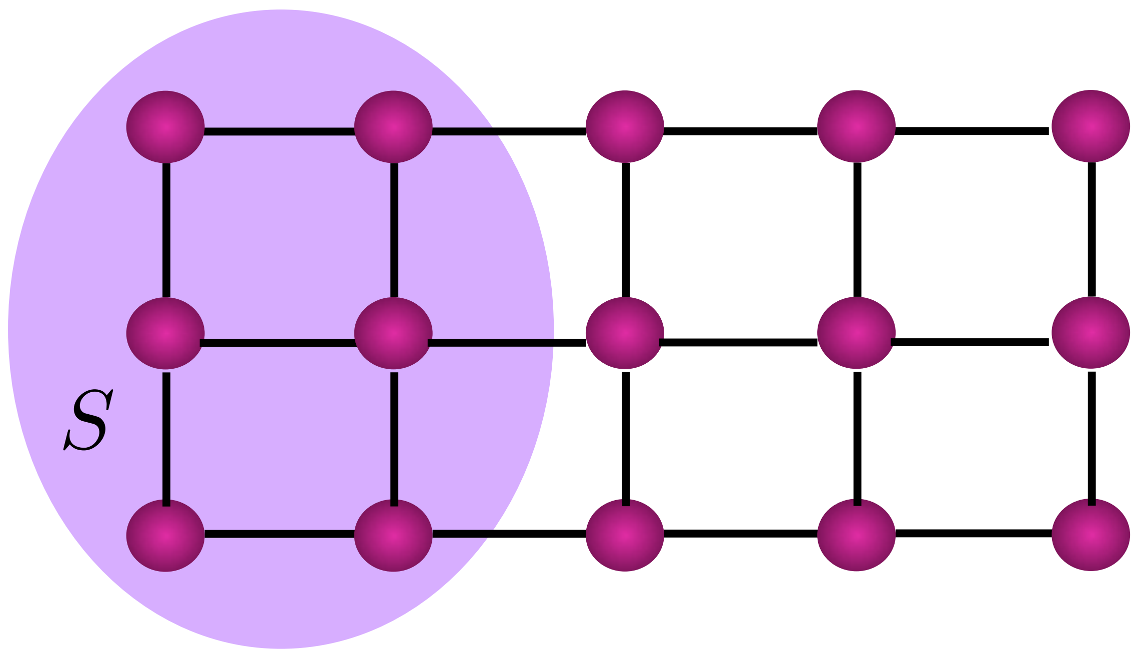
$$\Phi(S) = \frac{|\mathrm{bdry}(S)|}{\min\left(|S|, |V - S|\right)}$$



$$\Phi(S) = 1$$

Expansion

$$\Phi(S) = \frac{|\mathrm{bdry}(S)|}{\min\left(|S|, |V - S|\right)}$$



$$\Phi(S) = 1/2$$

Cheeger's Inequality and the Conjugate Gradient

Cheeger's inequality (degree-d unwted case)

$$\boxed{\frac{1}{2}\frac{\lambda_2}{d} \leq \frac{\Phi_G}{d} \leq \sqrt{2\frac{\lambda_2}{d}}}$$

λ_2 = second-smallest eigenvalue of L_G
~ d/mixing time of random walk

near d for expanders and random graphs

Cheeger's Inequality and the Conjugate Gradient

Cheeger's inequality (degree-d unwted case)

$$\boxed{\frac{1}{2}\frac{\lambda_2}{d} \leq \frac{\Phi_G}{d} \leq \sqrt{2\frac{\lambda_2}{d}}}$$

λ_2 = second-smallest eigenvalue of L_G

~ d/mixing time of random walk

Conjugate Gradient finds ϵ-approx solution to $L_G x = b$

in $O(\sqrt{d/\lambda_2}\log \epsilon^{-1})$ mults by L_G

is $O(dm\Phi_G^{-1}\log \epsilon^{-1})$ ops

Fast solution of linear equations

Conjugate Gradient fast when expansion high.

Elimination fast when low for G and all subgraphs.

Fast solution of linear equations

Conjugate Gradient fast when expansion high.

⇧

Planar graphs

⇩

Elimination fast when low for G and all subgraphs.

Problems:

Want speed of extremes in the middle

Fast solution of linear equations

Conjugate Gradient fast when expansion high.

Planar graphs

Elimination fast when low for G and all subgraphs.

Problems:

Want speed of extremes in the middle

Not all graphs fit into these categories!

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

ϵ-approx solution after
$O(\sqrt{\kappa(L_G, L_H)} \log \epsilon^{-1})$ iterations

condition number/approx quality

The relative condition number

$$\kappa(L_G, L_H) = \frac{\lambda_{max}(L_G L_H^+)}{\lambda_{min}(L_G L_H^+)}$$

pseudo-inverse

min non-zero eigenvalue

Inequalities and Approximation

$L_H \preccurlyeq L_G$ if for all x, $\quad x^T L_H x \preccurlyeq x^T L_G x$

Example: if H is a subgraph of G

$$\boldsymbol{x}^T L_G \boldsymbol{x} = \sum_{(u,v)\in E} w_{(u,v)} \left(\boldsymbol{x}(u) - \boldsymbol{x}(v)\right)^2$$

Inequalities and Approximation

$L_H \preccurlyeq L_G$ if for all x, $\quad x^T L_H x \preccurlyeq x^T L_G x$

$\kappa(L_G, L_H) \leq t$ if $\quad L_H \preccurlyeq L_G \preccurlyeq t L_H$

Call such an H a t-approx of G

Inequalities and Approximation

$L_H \preccurlyeq L_G$ if for all x, $\quad x^T L_H x \preccurlyeq x^T L_G x$

$\kappa(L_G, L_H) \leq t$ iff $\exists c : cL_H \preccurlyeq L_G \preccurlyeq ctL_H$

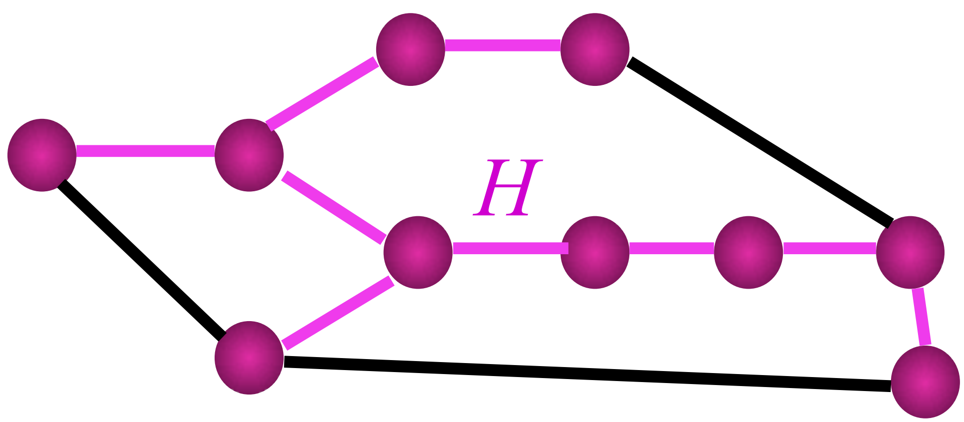
Call such an H a t-approx of G

Vaidya's Subgraph Preconditioners

Precondition G by a subgraph H

$L_H \preccurlyeq L_G$ so just need t for which $L_G \preccurlyeq t L_H$

Easy to bound t if H is a spanning tree



And, easy to solve equations in L_H by elimination

Approximate Laplacian Solvers

Preconditioned Conjugate Gradient $O(mn)$
[Hestenes '51, Stiefel '52, ???]

Vaidya '90: Augmented MST $O(mn^{3/4})$

Boman-Hendrickson '01:
Using Low-Stretch Spanning Trees $\tilde{O}(mn^{1/2})$

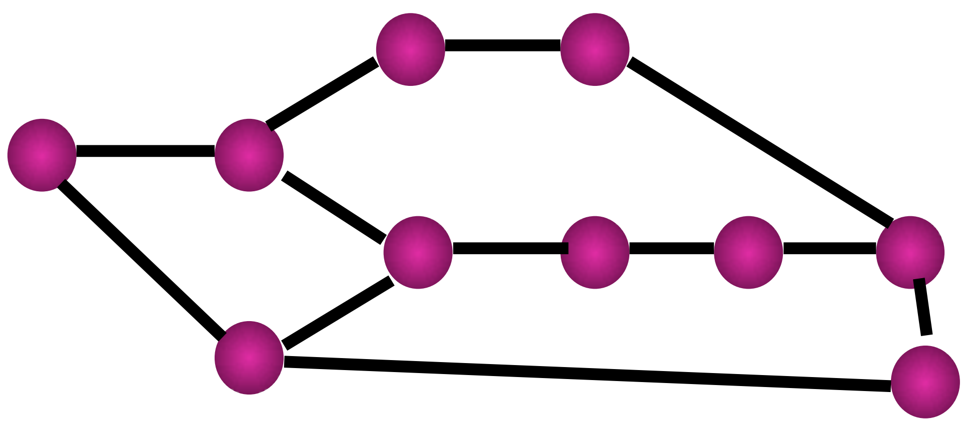
S-Teng '04: Spectral sparsification $O(m \log^c n)$

Koutis-Miller-Peng '11: Elegance $\tilde{O}(m \log n)$

The Stretch of Spanning Trees

Boman-Hendrickson '01: $L_G \preccurlyeq \mathrm{st}_G(T) L_T$

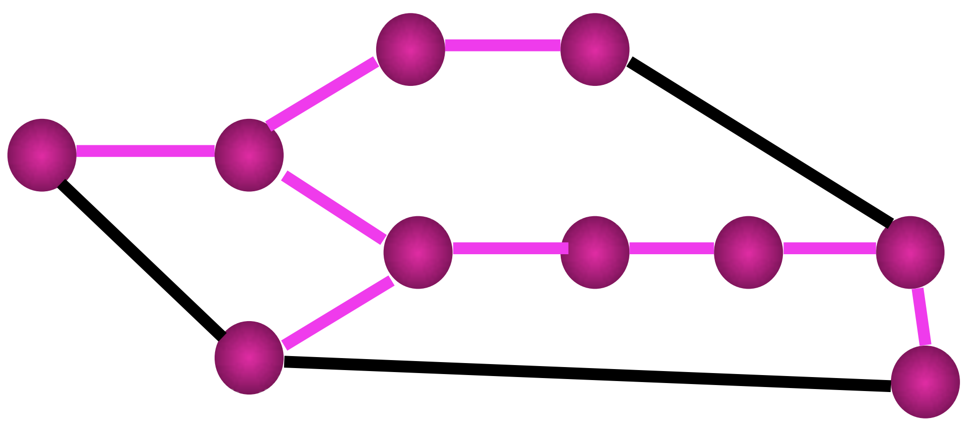
Where $\mathrm{st}_T(G) = \sum_{(u,v) \in E} \text{path-length}_T(u,v)$



The Stretch of Spanning Trees

Boman-Hendrickson '01: $L_G \preccurlyeq \mathrm{st}_G(T) L_T$

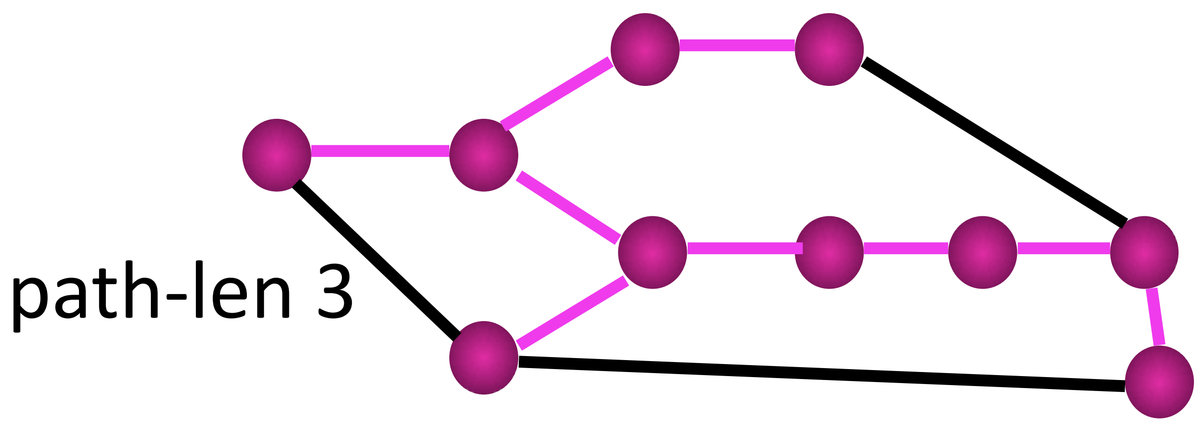
Where $\mathrm{st}_T(G) = \sum_{(u,v) \in E} \text{path-length}_T(u, v)$



The Stretch of Spanning Trees

Boman-Hendrickson '01: $L_G \preccurlyeq \mathrm{st}_G(T) L_T$

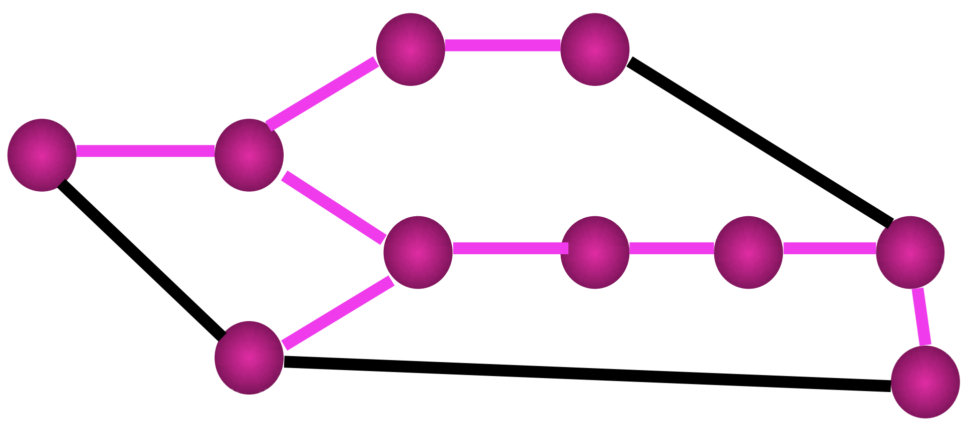
Where $\mathrm{st}_T(G) = \sum_{(u,v) \in E} \text{path-length}_T(u,v)$



The Stretch of Spanning Trees

Boman-Hendrickson '01: $L_G \preccurlyeq \mathrm{st}_G(T) L_T$

Where $\mathrm{st}_T(G) = \sum_{(u,v)\in E} \text{path-length}_T(u,v)$

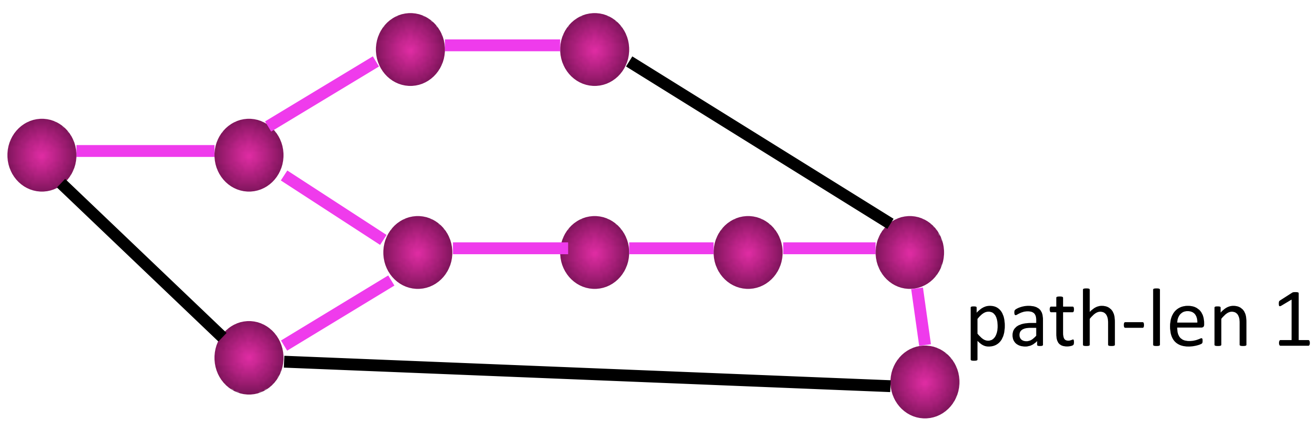


path-len 5

The Stretch of Spanning Trees

Boman-Hendrickson '01: $L_G \preccurlyeq \mathrm{st}_G(T) L_T$

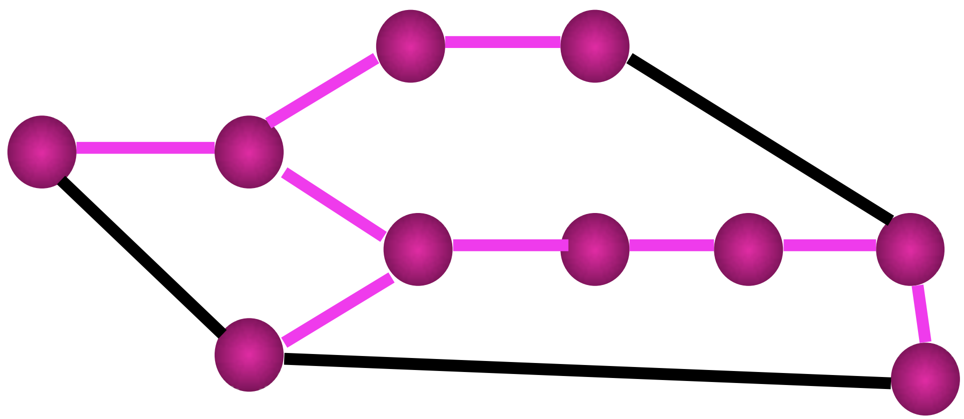
Where $\mathrm{st}_T(G) = \sum_{(u,v)\in E} \text{path-length}_T(u,v)$



The Stretch of Spanning Trees

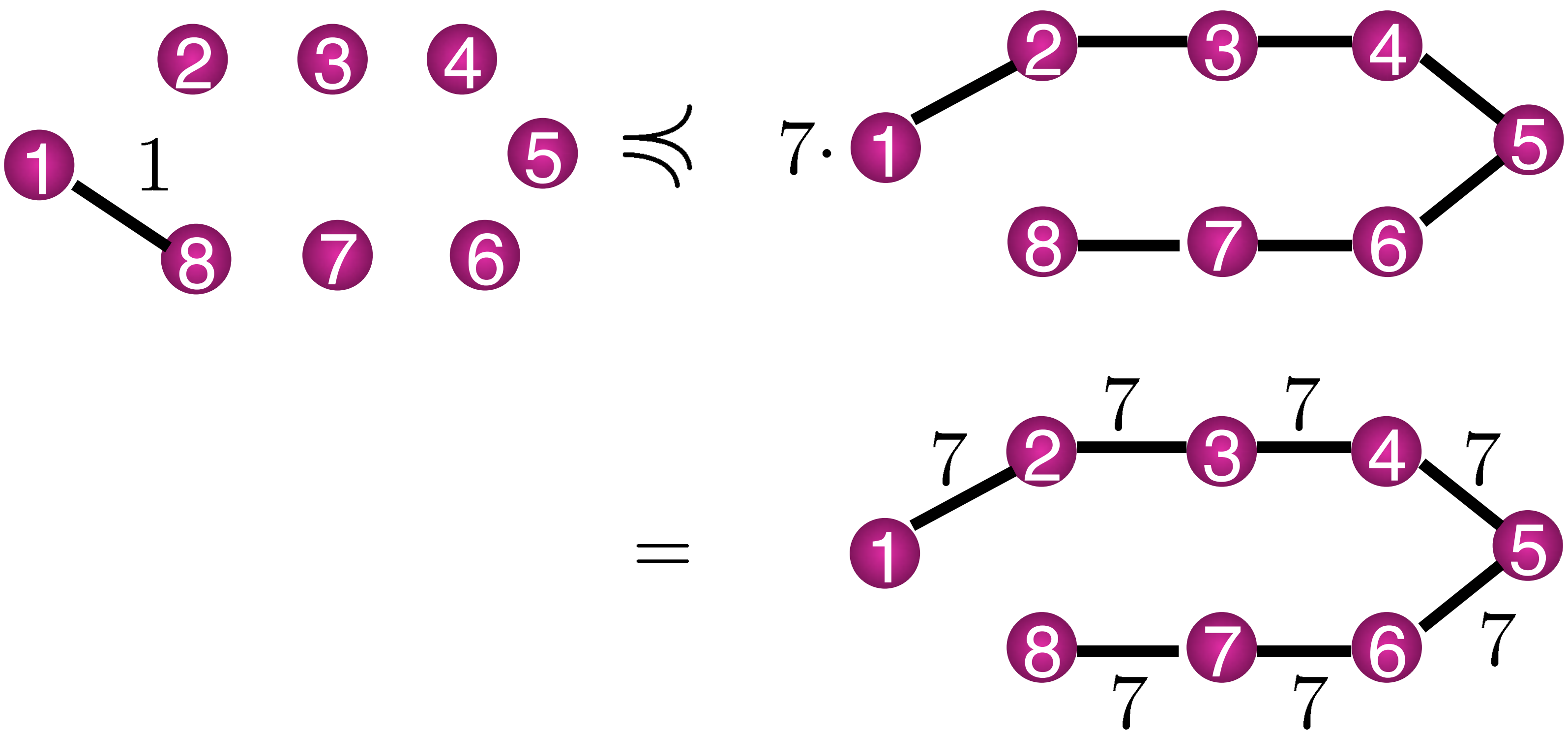
Boman-Hendrickson '01: $L_G \preccurlyeq \mathrm{st}_G(T) L_T$

Where $\mathrm{st}_T(G) = \sum_{(u,v) \in E} \text{path-length}_T(u, v)$



In weighted case, measure resistances of paths

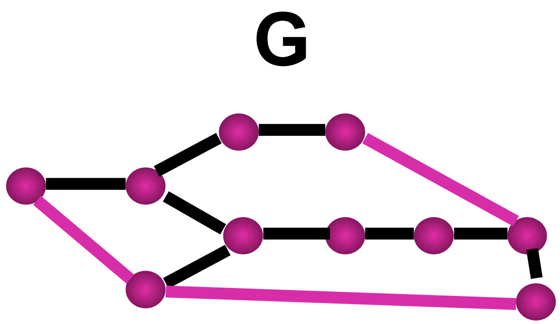
Fundamental Graphic Inequality



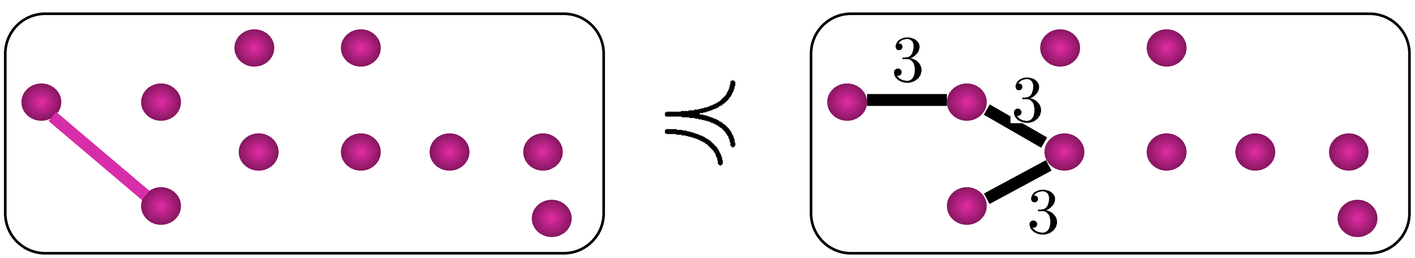
edge $\preccurlyeq$ k times path of length k

With weights, corresponds to resistors in serial
(Poincaré inequality)

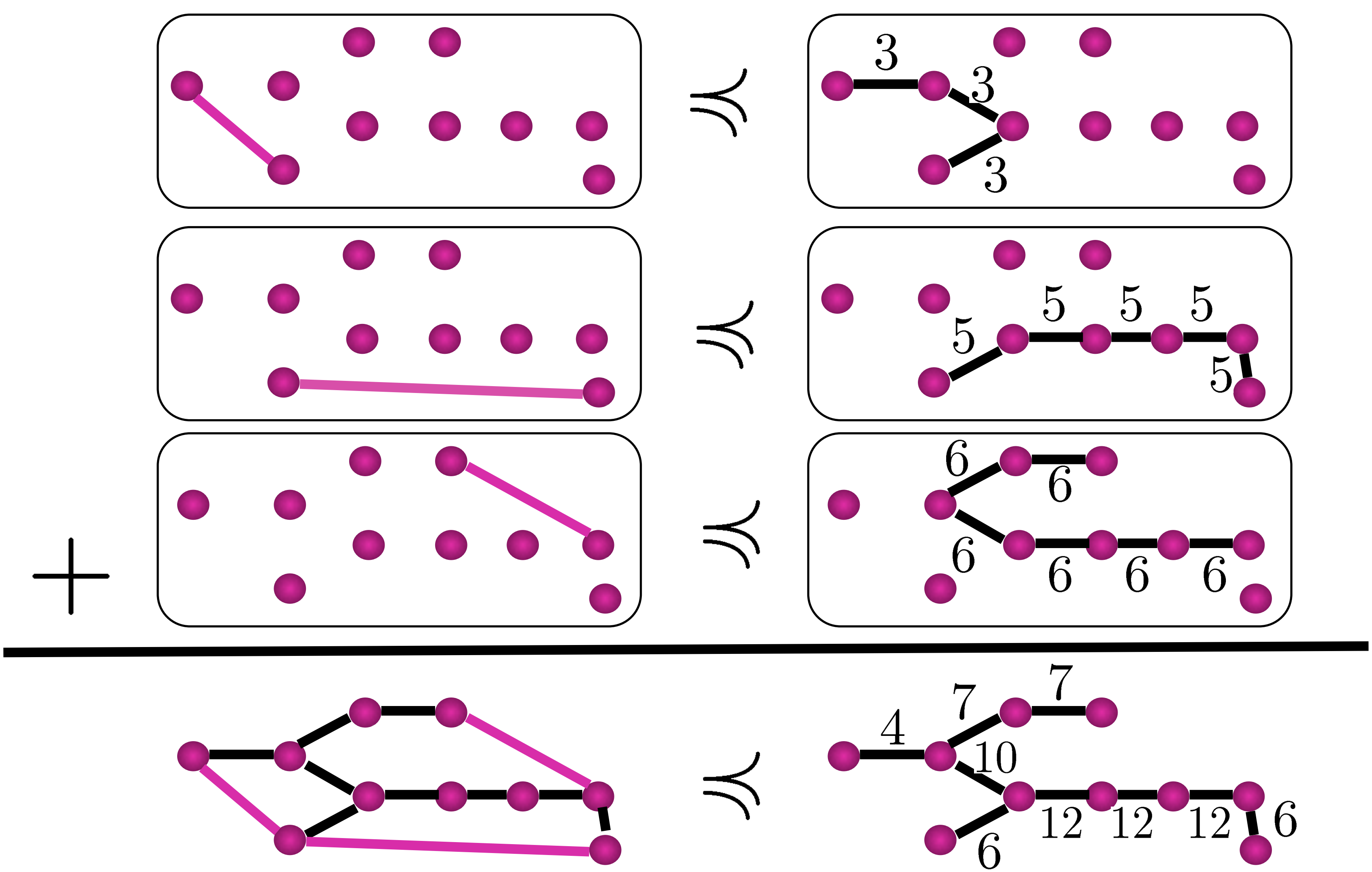
When T is a Spanning Tree



Every edge of G not in T has unique path in T



When T is a Spanning Tree



Low-Stretch Spanning Trees

For every G there is a T with

$$\mathrm{st}_T(G) \leq m^{1+o(1)}$$ where $m = |E|$

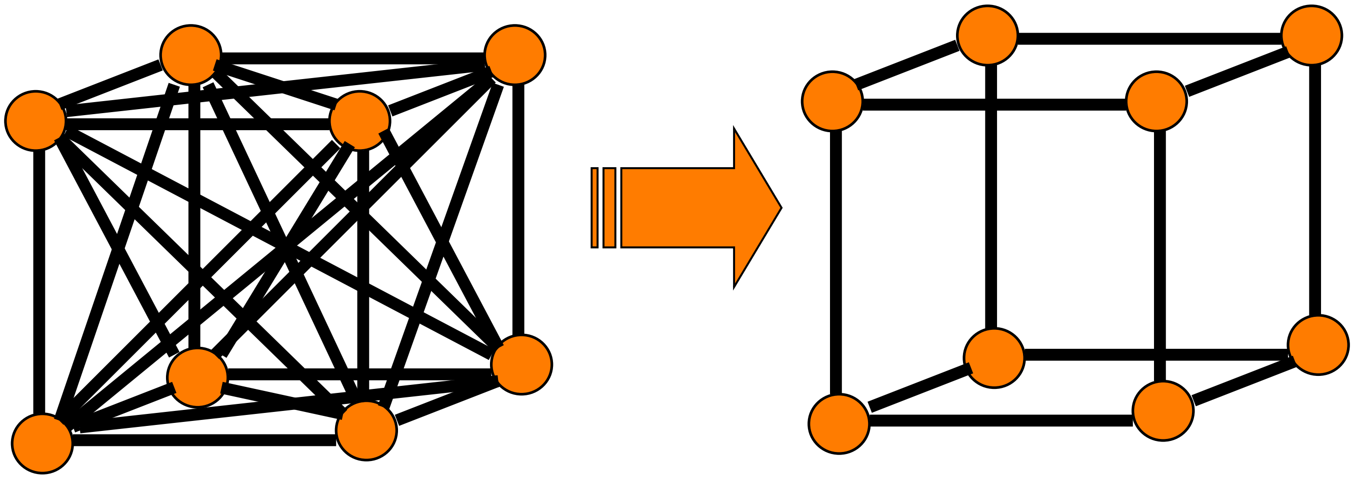
(Alon-Karp-Peleg-West '91)

$$\mathrm{st}_T(G) \leq O(m \log m \log^2 \log m)$$

(Elkin-Emek-S-Teng '04, Abraham-Bartal-Neiman '08)

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

Spectral Sparsification [S-Teng '04]

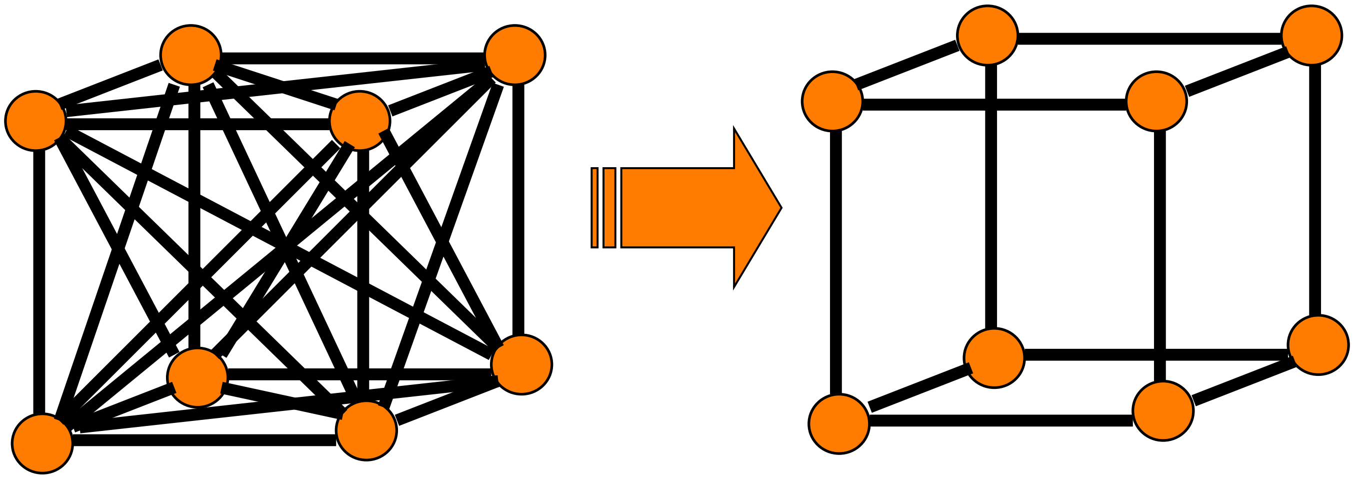


Approximate G by a sparse H with

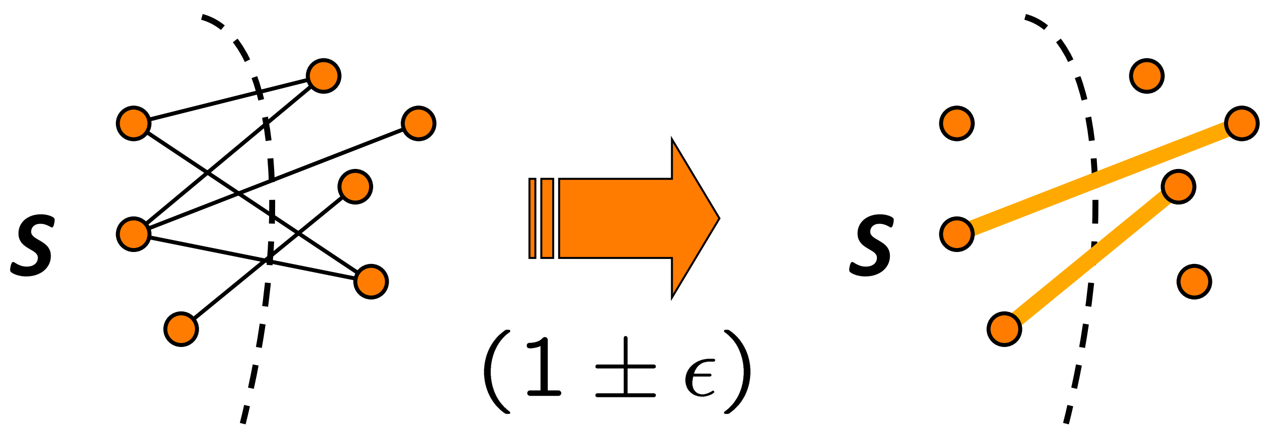
$$\kappa(L_G, L_H) \leq 1 + \epsilon$$

$$v^T L_G v \xrightarrow[(1 \pm \epsilon)]{} v^T L_H v$$

Cut Sparsification [Benczur-Karger '96]



Approximate G by a sparse H,
approximately preserving all boundaries



Sparsification

Goal: find sparse approximation for every G

S-Teng '04: For every G is an H with
$O(n \log^7 n/\epsilon^2)$ edges and $\kappa(L_G, L_H) \leq 1+\epsilon$

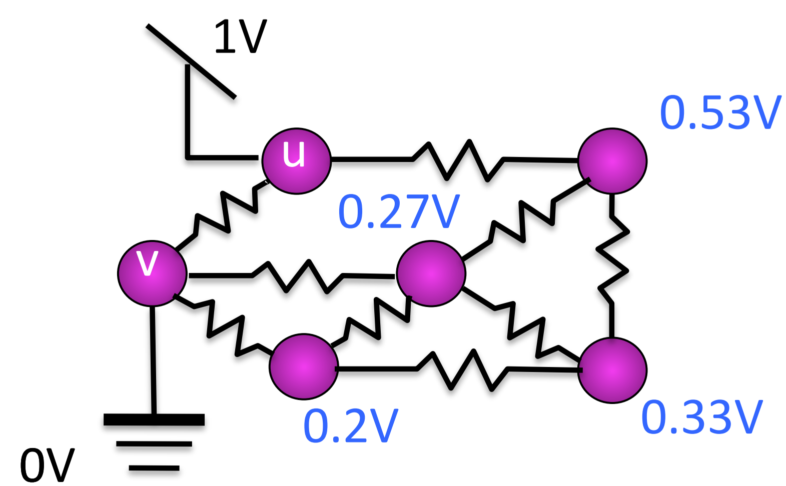
Sparsification

Goal: find sparse approximation for every G

S-Teng '04: For every G is an H with
$O(n \log^7 n/\epsilon^2)$ edges and $\kappa(L_G, L_H) \leq 1 + \epsilon$

S-Srivastava '08: with $O(n \log n/\epsilon^2)$ edges
by random sampling by effective resistances

1/(current flow at one volt)



Sparsification

Goal: find sparse approximation for every G

S-Teng '04: For every G is an H with

$O(n \log^7 n/\epsilon^2)$ edges and $\kappa(L_G, L_H) \leq 1 + \epsilon$

S-Srivastava '08: with $O(n \log n/\epsilon^2)$ edges

Batson-S-Srivastava '09

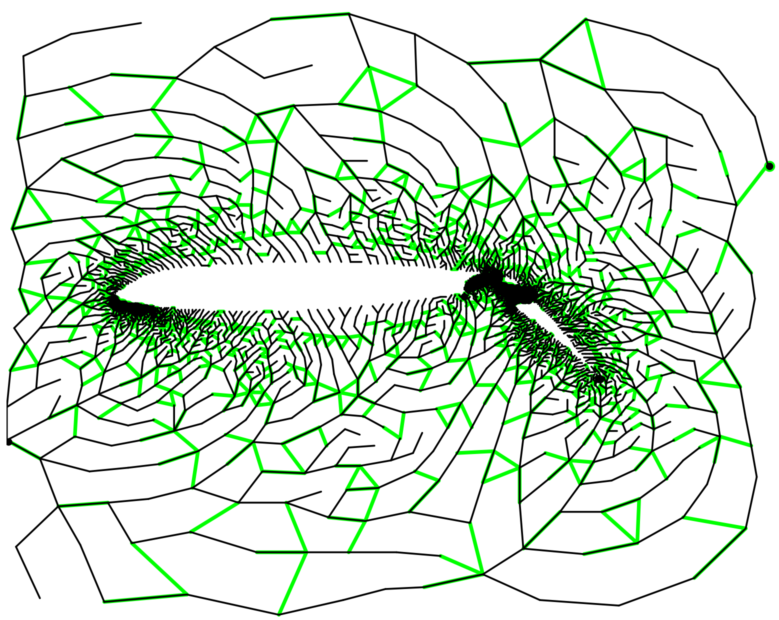
deterministic, poly time, and $O(n/\epsilon^2)$ edges

Sparsifiers

Low-Stretch Trees

Ultra-Sparsifiers [S-Teng]

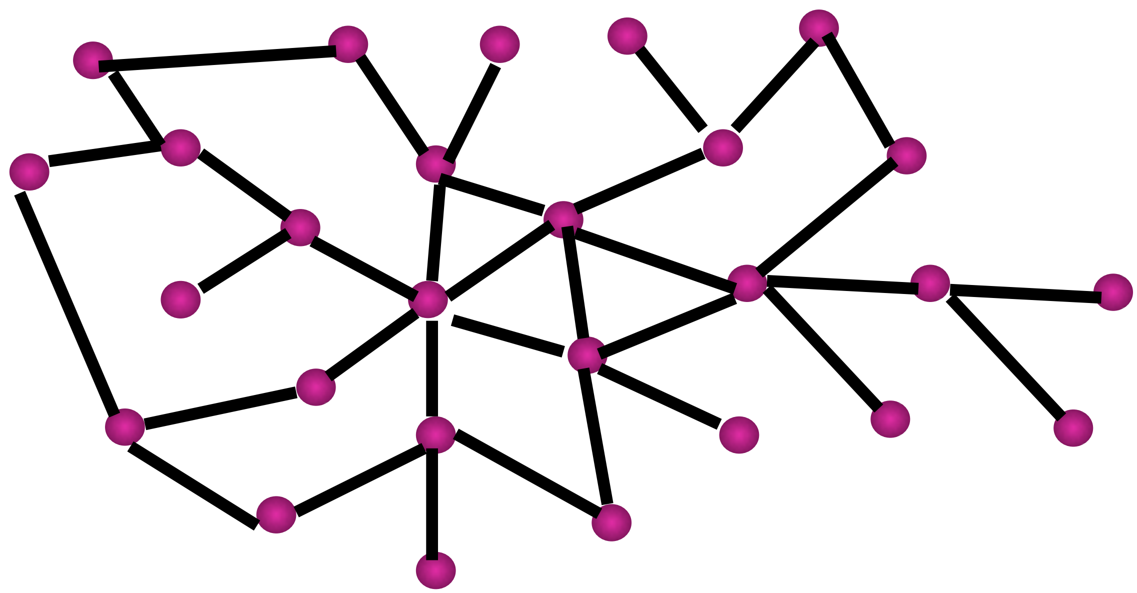
Approximate G by a tree plus $n/\log^2 n$ edges



$$L_H \preccurlyeq L_G \preccurlyeq c\log^2 n \; L_H$$

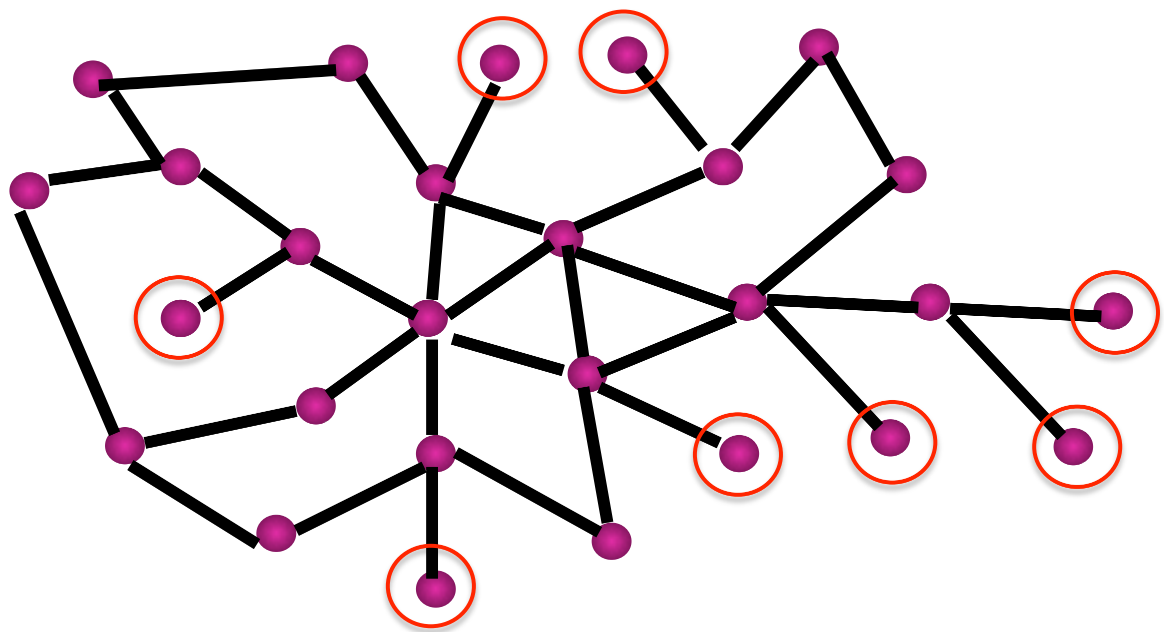
Cholesky factor to smaller system

Eliminate degree 1 and 2 nodes



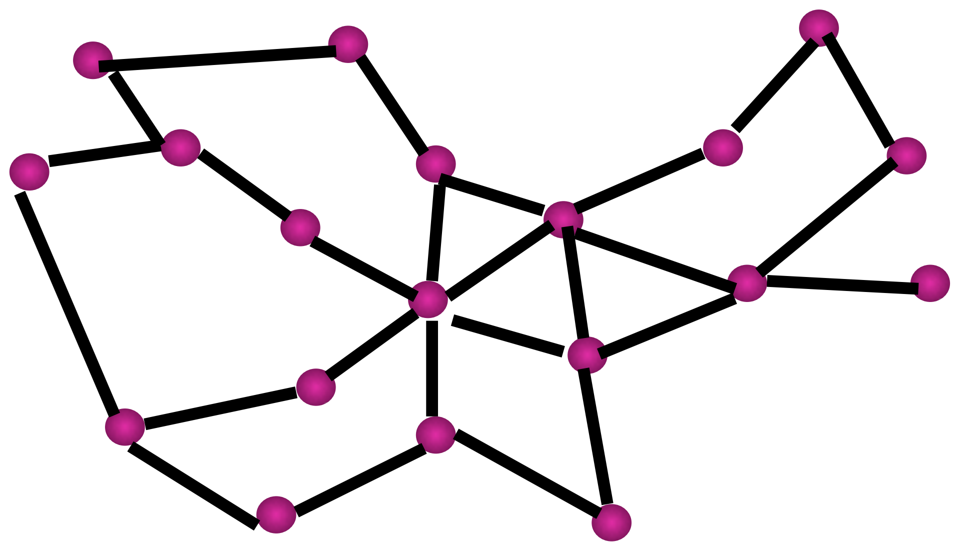
Cholesky factor to smaller system

Eliminate degree 1 and 2 nodes



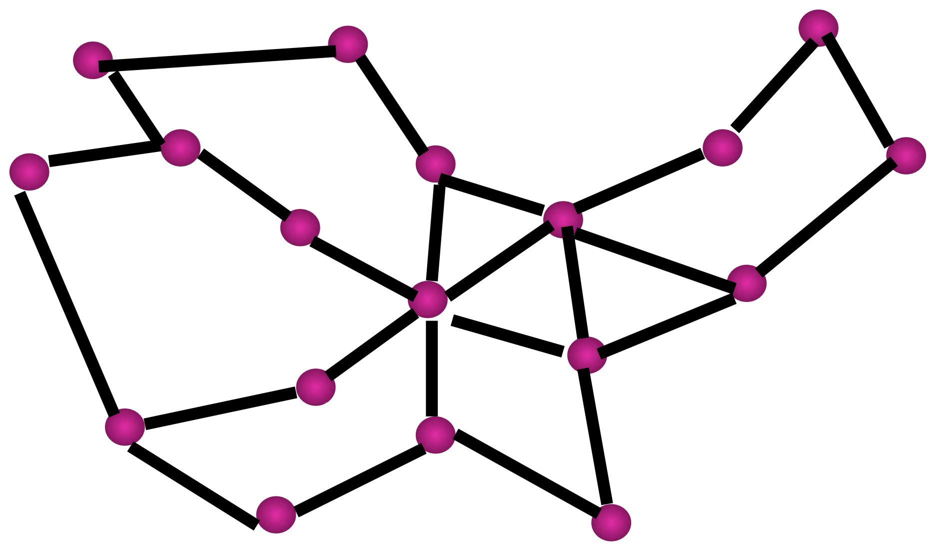
Cholesky factor to smaller system

Eliminate degree 1 and 2 nodes



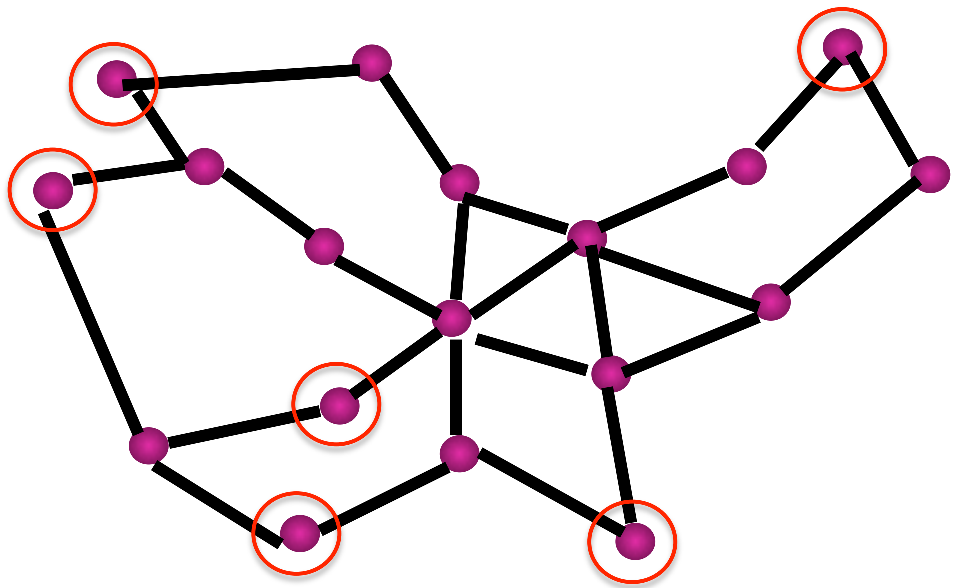
Cholesky factor to smaller system

Eliminate degree 1 and 2 nodes



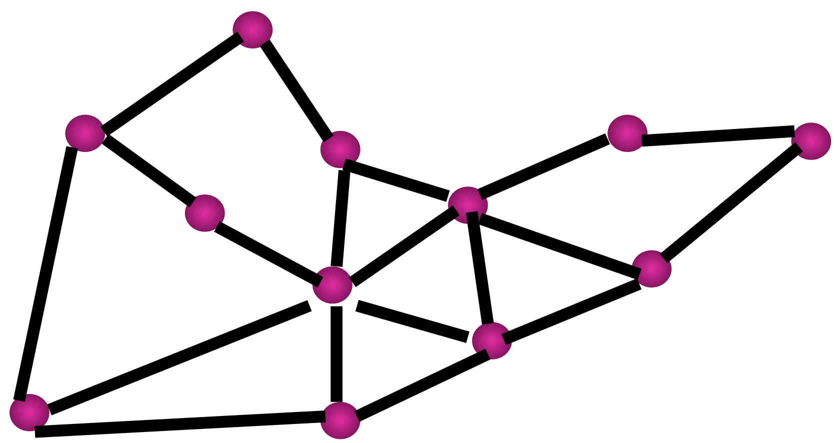
Cholesky factor to smaller system

Eliminate degree 1 and 2 nodes



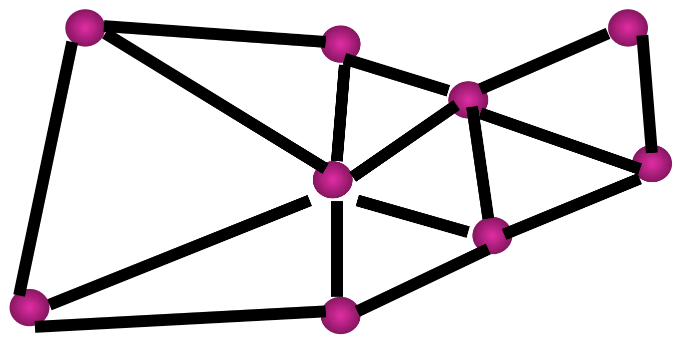
Cholesky factor to smaller system

Eliminate degree 1 and 2 nodes



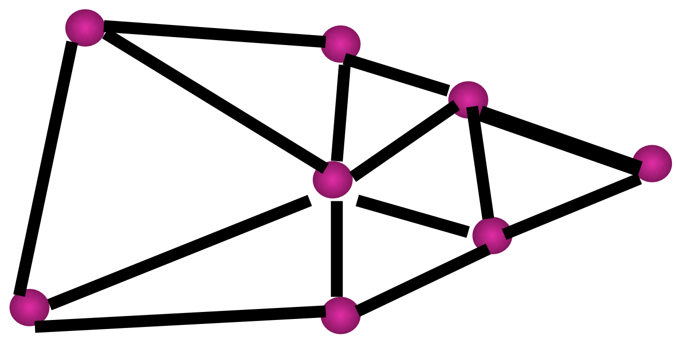
Cholesky factor to smaller system

Eliminate degree 1 and 2 nodes



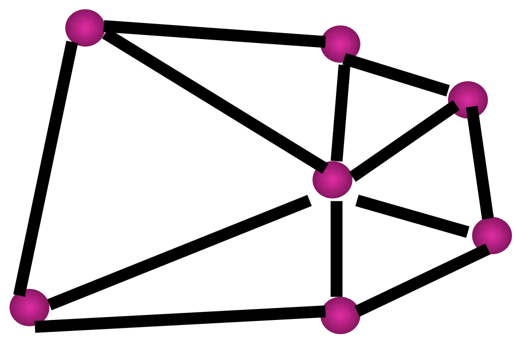
Cholesky factor to smaller system

Eliminate degree 1 and 2 nodes



Cholesky factor to smaller system

Eliminate degree 1 and 2 nodes



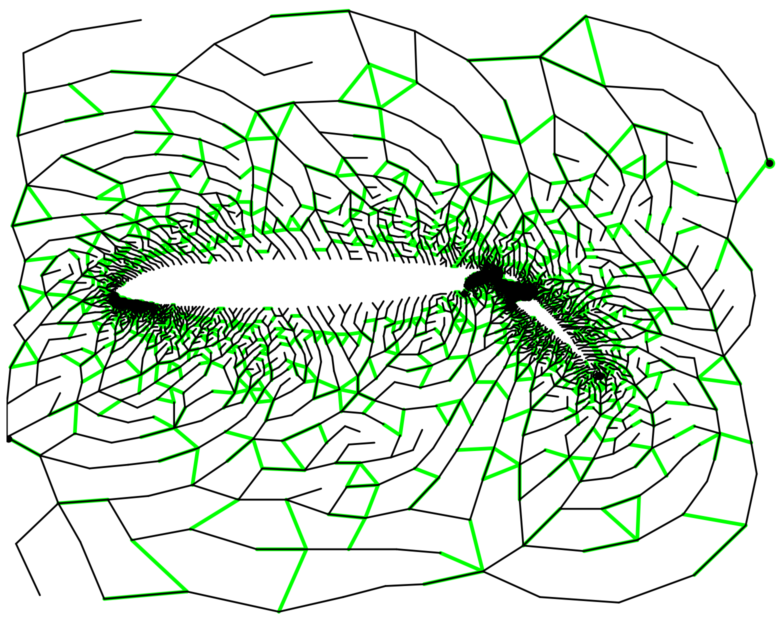
Get system of size $O(n/\log^2 n)$, solve recursively

[Joshi '97, Reif '98, S-Teng '04 '09]

Ultra-Sparsifiers

Solve systems in H by:

1. Cholesky eliminating degree 1 and 2 nodes
2. recursively solving reduced system



Time

$$O(m \log^c m)$$

Koutis-Miller-Peng '11

Solve in time $O(m \log n \ \log^2 \log n \ \log(1/\epsilon))$

Build Ultra-Sparsifier by:

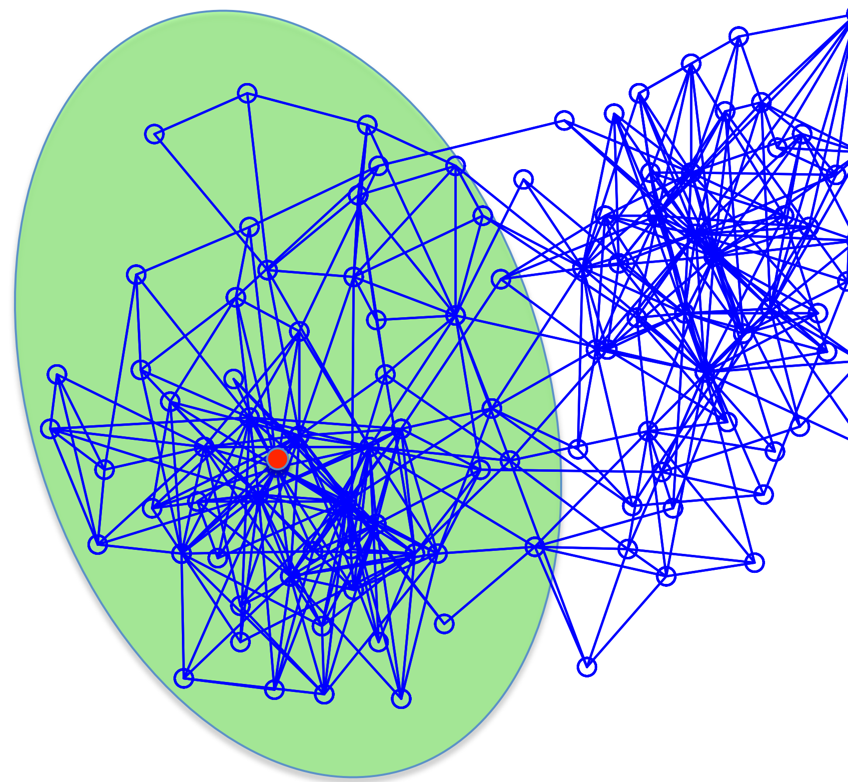
1. Constructing low-stretch spanning tree
2. Adding other edges with probability

$$p_{u,v} \sim \text{path-length}_T(u,v)$$

Code by Yiannis Koutis

Local Graph Clustering [S-Teng '04]

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



See algorithms of
Andersen-Chung-Lang '06 and
Andersen-Peres '09

Open Problems

Faster and better Low-Stretch Spanning Trees.

Faster high-quality sparsification.

Faster local clustering and graph decomposition.

Other families of linear systems
from physical problems
from optimization

Conclusions

Laplacian Solvers are a powerful primitive!

Faster Maxflow: Christiano-Kelner-Madry-S-Teng

Faster Random Spanning Trees: Kelner-Madry-Propp

All Effective Resistances: S-Srivastava

Maybe we can solve all well-conditioned graph problems in nearly-linear time.

Don't fear large constants