Spectral Graph Sparsification

Sushant Sachdeva
University of Toronto

Simons Fall 2023 Bootcamp

What We’ll See Today

[Spielman-Srivastava ‘08]
A linear algebraic approach to sparsifying graphs while preserving cut structure

Recall

Positive Semi-Definite Matrices

A symmetric matrix A is psd (positive semi-definite) iff

$$\forall x \quad x^\top A x \geq 0$$

Laplacians are psd

Approximation for PSD Matrices

Define $A \approx_\varepsilon B$ iff

$$\forall x \quad e^{-\varepsilon} \cdot x^\top B x \leq x^\top A x \leq e^{\varepsilon} \cdot x^\top B x$$

Equivalently,

$$e^{-\varepsilon} \cdot B \preccurlyeq A \preccurlyeq e^{\varepsilon} \cdot B$$

Approximation for PSD Matrices

The relation $\approx_\varepsilon$ satisfies

1. Reflexive $A \approx_\varepsilon A$
2. Symmetric $A \approx_\varepsilon B$ implies $B \approx_\varepsilon A$
3. Additivity, $A_1 \approx_\varepsilon B_1$and $A_2 \approx_\varepsilon B_2$ implies $A_1 + A_2 \approx_\varepsilon B_1 + B_2$
4. "Transitive"$A \approx_\varepsilon C$ and $B \approx_\delta A$, imply $A \approx_{\varepsilon+\delta} C$

Spectral Graph Sparsifiers

Suppose G, H are graphs on the same vertex set.

If $L_G \approx_\varepsilon L_H$

$$\forall x \quad e^{-\varepsilon} \cdot x^\top L_H x \le x^\top L_G x \le e^{\varepsilon} \cdot x^\top L_H x$$

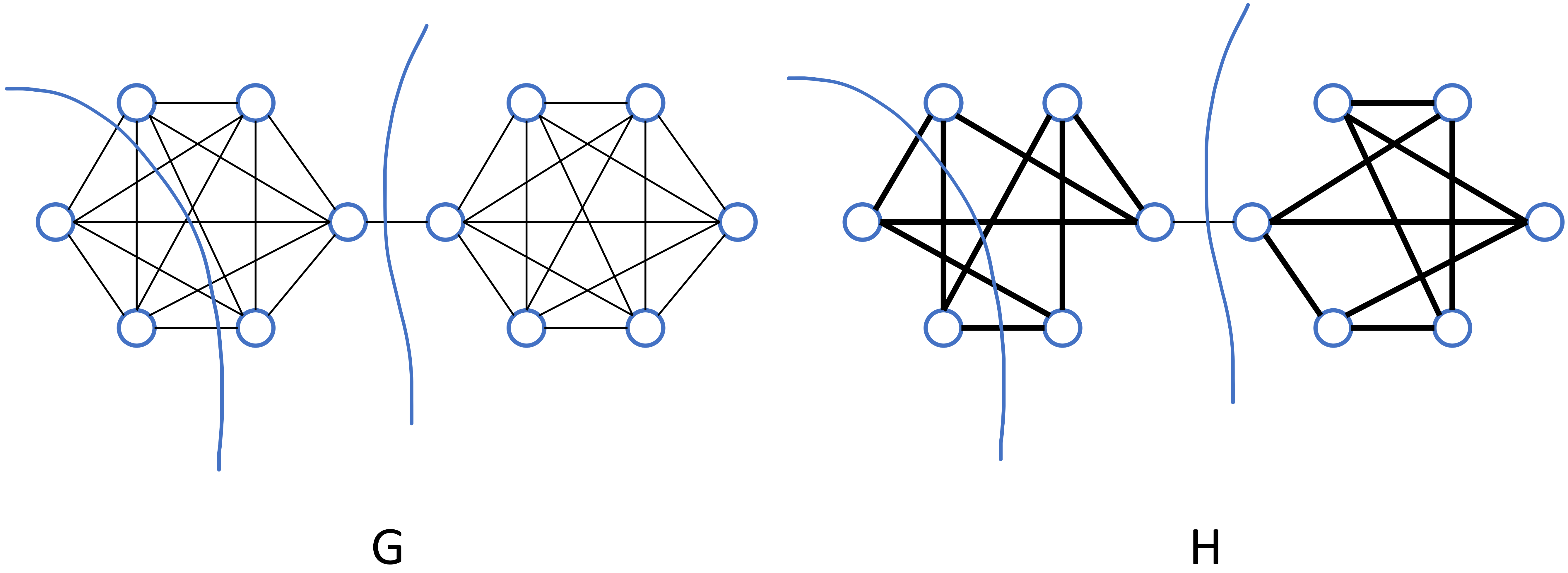
Picking $x = \mathbf{1}_S$,

$$e^{-\varepsilon} \cdot w_H(E(S, S^c)) \le w_G(E(S, S^c)) \le e^{\varepsilon} \cdot w_H(E(S, S^c))$$

All vertex cuts have approximately equal weights in G and H

Generalizes cut-sparsification [Benczur-Karger '92]

Cut Sparsifiers



Spectral Graph Sparsifiers

[Spielman-Srivastava '08]
For every graph G, we can find a re-weighted subgraph H with $O(n\varepsilon^{-2}\log n)$ edges such that $L_G \approx_\varepsilon L_H$

Approach:

1. Define a random graph H s.t. $\mathbb{E}L_H = L_G$
2. Prove using "matrix concentration" that $L_H \approx_\varepsilon L_G$

Setting Expectations

Recall

$$L_G = \sum_{e \in E} w_e L_e$$

We are sampling a reweighted subgraph

$$L_H = \sum_{e \in E} w'_e L_e$$

Setting Expectations

Toss an independent coin for edge e with prob. p_e
If heads, include edge e with weight w'_e,
If tails, discard

$$\mathbb{E}L_H = \sum_{e \in E} p_e w'_e L_e$$

$w'_e = \frac{w_e}{p_e}$ gives

$$\mathbb{E}L_H = \sum_{e \in E} w_e L_e = L_G$$

Scalar Chernoff Bounds

Consider random variables $X_i \in R$ s.t

$$0 \leq X_i \leq B$$

And $\sum_i \mathbb{E}X_i = 1$. Then,

$$\Pr\left[\sum_i X_i \approx_\varepsilon 1\right] \geq 1 - 2e^{-\varepsilon^2/4B}$$

Matrix Chernoff Bounds

[Tropp '11]
Consider symmetric random matrices $X_i \in R^{n\times n}$ s.t

$$0 \preccurlyeq X_i \preccurlyeq B\mathbb{I}$$

And $\sum_i \mathbb{E}X_i = \mathbb{I}$. Then,

$$\Pr\left[\sum_i X_i \approx_\varepsilon \mathbb{I}\right] \geq 1 - 2ne^{-\varepsilon^2/4B}$$

Matrix Chernoff Bounds

[Tropp '11]

Consider symmetric random matrices $X_i \in R^{n\times n}$ s.t

$$0 \preccurlyeq X_i \preccurlyeq \frac{\varepsilon^2}{4c\ln 2n}\mathbb{I}$$

And $\sum_i \mathbb{E}X_i = \mathbb{I}$. Then,

$$\Pr\left[\sum_i X_i \approx_\varepsilon \mathbb{I}\right] \geq 1 - \frac{1}{n^{c-1}}$$

Red variables are in isotropic position $X = L_G^{-1/2} X_i L_G^{-1/2}$

Applying Matrix Chernoff

Our matrix random variables

$$X_i = \begin{cases} \frac{w_e}{p_e} L_e & with\ prob\ p_e \\ 0 & with\ prob\ 1 - p_e \end{cases}$$

We want $\sum_i \mathbb{E} X_i = \mathbb{I}$. We have $\sum_i \mathbb{E} X_i = L_G$.

Equivalently, $\sum_i L_G^{-1/2} X_i L_G^{-1/2} = \mathbb{I}$

Consider variables in isotropic position $X = L_G^{-1/2} X_i L_G^{-1/2}$

Red variables are in isotropic position $X = L_G^{-1/2} X_i L_G^{-1/2}$

Applying Matrix Chernoff

Our matrix random variables

$$X_i = \begin{cases} \dfrac{w_e}{p_e} L_e & with\ prob\ p_e \\ 0 & with\ prob\ 1 - p_e \end{cases}$$

We have

$$\sum_i \mathbb{E} X_i = \mathbb{I}$$

Remains to achieve $||X_i|| \lesssim \dfrac{\varepsilon^2}{\log n}$

Red variables are in isotropic position $X = L_G^{-1/2} X_i L_G^{-1/2}$

Applying Matrix Chernoff

Our matrix random variables

$$X_i = \begin{cases} \frac{w_e}{p_e} L_e & \text{with prob } p_e \\ 0 & \text{with prob } 1 - p_e \end{cases}$$

$$\|X_i\| = \frac{w_e}{p_e} \left\| L_G^{-1/2} L_e L_G^{-1/2} \right\|$$

$$= \frac{w_e}{p_e} \left\| L_G^{-1/2} (1_u - 1_v)(1_u - 1_v)^\top L_G^{-1/2} \right\|$$

$$= \frac{w_e}{p_e} (1_u - 1_v)^\top L_G^{-1} (1_u - 1_v)$$

Achieving Small Norm

$$R_G(u, v) = (1_u - 1_v)^\top L_G^{-1}(1_u - 1_v)$$

Effective resistance across (u, v).

Voltage difference across (u, v) to achieve 1 unit current

Achieving Small Norm

Suffices to pick

$$\frac{w_e}{p_e} R_G(e) = \frac{\varepsilon^2}{\log n}$$

i.e.

$$p_e = \frac{\log n}{\varepsilon^2} \cdot w_e R_G(e)$$

$w_e R_G(e)$ is called leverage score of edge e

Expected number of Edges

Expected number of edges with >0 weight

$$\sum_e p_e = \frac{\log n}{\varepsilon^2} \cdot \sum_e w_e R_G(u,v)$$

$$= \frac{\log n}{\varepsilon^2} \cdot \sum_e w_e Tr((1_u - 1_v)^\top L_G^{-1}(1_u - 1_v))$$

$$= \frac{\log n}{\varepsilon^2} \cdot Tr\left(L_G^{-1} \sum_e w_e (1_u - 1_v)(1_u - 1_v)^\top\right)$$

$$= \frac{\log n}{\varepsilon^2} \cdot Tr(L_G^{-1} L_G\ ) \leq n\varepsilon^{-2} \log n$$

Caveat

$\frac{\log n}{\varepsilon^2} \cdot w_e R_G(u, v)$ can be larger than 1

Trick: Splitting an edge into k parallel edges, with weight $\frac{w_e}{k}$ each

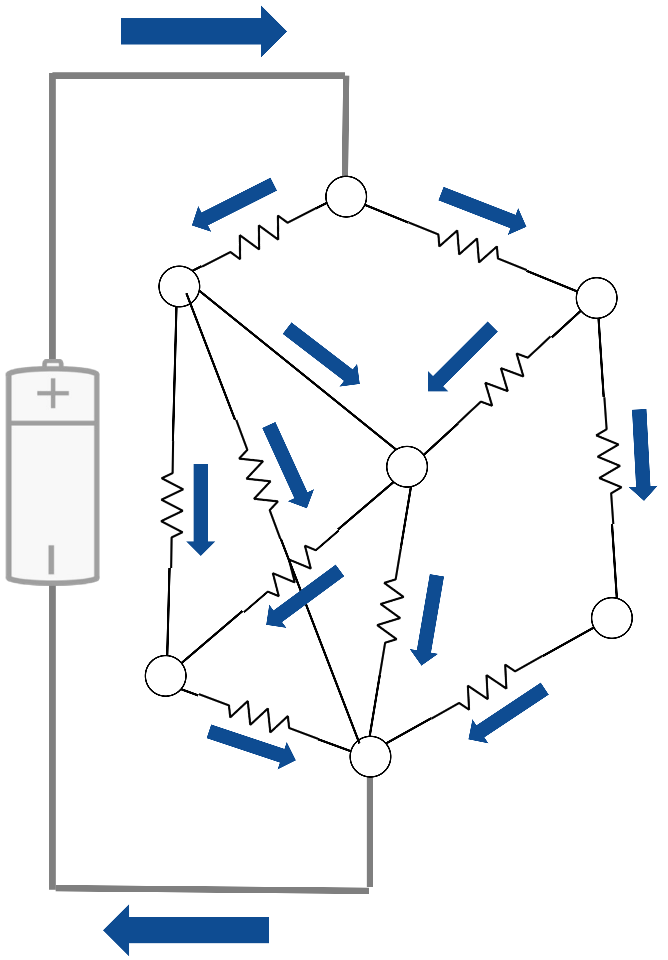
Also works, but slightly different proof:

Keep all edges with

$$\frac{\log n}{\varepsilon^2} \cdot w_e R_G(u, v) \geq 1$$

Fast Laplacian Solvers

Solving a Laplacian System



$$v = L^{-1}d$$

Applications of Laplacian Solvers

PDEs via Finite Element Method [Str86, BHV08]
Interior Point Methods for Optimization [KRS15, CMSV16]
Learning on graphs [ZGL03, ZS04, ZBLWS04]
Faster flow algorithms [DS08, CKMST11, KMP12, Mad13, LS14, LS20, AMV20, BLNPSSSW20, KLS21, BLLSSSW21, DGGLPSY22]
Graph partitioning [OSV12]
Sampling random spanning trees [KM09, MST15, DKPRS17]
Graph sparsification [SS08, LKP12]

[Spielman-Teng ‘04] Laplacian linear systems $Lx = d$ can be solved in nearly linear time

Gödel Prize 2015

Approximately Solving a System

A δ-approximate solution to $Lx = d$
is $\tilde{x}$ s.t.

$$\|\tilde{x} - L^{-1}d\|_L \leq \delta \|x\|_L$$

Where $\|x\|_L = \sqrt{x^\top L x}$

This is the right norm for most applications!

Lots of Improved Algorithms

[Spielman-Teng ‘04] $m \log^{O(1)} n$

Several improvements [KMP10, KMP11, KOSZ13, CKMPPRX14, PS14, LPS13, KLPSS16, KS16, JS21] $m(\log \log n)^{O(1)}$

What We'll See Today

[S-Zhao SPAA '23]

A very simple algorithm to solve Laplacian systems in $m\log^4 n$ time

Also a parallel algorithm with $\log^2 n$ depth

Solving Well-Conditioned Systems

To solve $Ax = b$ approximately, where $A \approx_{0.5} \mathbb{I}$

$$x^{(i+1)} \leftarrow x^{(i)} - \left(Ax^{(i)} - b\right)$$

$$x^{(i+1)} - x^\star = (\mathbb{I} - A)\left(x^{(i)} - x^\star\right)$$
$$\left\|x^{(i+1)} - x^\star\right\|_A \leq \|\mathbb{I} - A\| \cdot \left\|x^{(i)} - x^\star\right\|_A$$

Starting with $x^{(0)} = 0$, $\left\|x^{(t)} - x^\star\right\|_A \leq 0.9^{-t} \cdot \|x^\star\|_A$

δ-approximate solution in $O\left(\log\frac{1}{\delta}\right)$ iterations

Iterative Refinement

To solve $Ax = b$ approximately,

Find C easy to invert AND $A \approx_{0.5} C$

Solve $C^{-1}Ax = C^{-1}b$ by

$$x^{(i+1)} \leftarrow x^{(i)} - \left(C^{-1}Ax^{(i)} - C^{-1}b\right)$$

δ-approximate solution in $O\left(\log\frac{1}{\delta}\right)$ iterations

Block Cholesky Factorization

Block Cholesky Factorization

$$\begin{array}{cc} & \begin{array}{cc} F & T \end{array} \\ L = & \begin{pmatrix} A & B \\ B^\top & C \end{pmatrix} \end{array} \begin{array}{c} \\ F \\ T \end{array}$$

$$V = F \cup T$$

Block Cholesky Factorization

$$L = \begin{array}{cc} & \begin{array}{cc} F & T \end{array} \\ & \begin{pmatrix} A & B \\ B^\top & C \end{pmatrix} \begin{array}{c} F \\ T \end{array} \end{array} \qquad V = F \cup T$$

$$L = \begin{pmatrix} I & 0 \\ B^\top A^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & SC(L,T) \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix}$$

Schur Complement: $SC(L,T) = C - B^\top A^{-1} B$

Block Cholesky Factorization

$$L = \begin{array}{cc} & \begin{array}{cc} F & T \end{array} \\ & \begin{pmatrix} A & B \\ B^\top & C \end{pmatrix} \begin{array}{c} F \\ T \end{array} \end{array} \qquad V = F \cup T$$

$$L = \begin{pmatrix} I & 0 \\ B^\top A^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & SC(L,T) \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix}$$

Schur Complement: $SC(L,T) = C - B^\top A^{-1} B$

Fact: Schur complement of a Laplacian is also Laplacian

Block Cholesky Factorization

$$L = \begin{array}{cc} & \begin{array}{cc} F & T \end{array} \\ & \begin{pmatrix} A & B \\ B^\top & C \end{pmatrix} \begin{array}{c} F \\ T \end{array} \end{array} \qquad V = F \cup T$$

$$L^+ = \begin{pmatrix} I & -A^{-1}B \\ 0 & I \end{pmatrix} \begin{pmatrix} A^{-1} & 0 \\ 0 & SC(L,T)^+ \end{pmatrix} \begin{pmatrix} I & 0 \\ -B^\top A^{-1} & I \end{pmatrix}$$

Schur Complement: $SC(L,T) = C - B^\top A^{-1} B$

Block Cholesky Factorization

$$L = \begin{array}{cc} & \begin{array}{cc} F & T \end{array} \\ & \begin{pmatrix} A & B \\ B^\top & C \end{pmatrix} \begin{array}{c} F \\ T \end{array} \end{array}$$

$$V = F \cup T$$

$$L^+ = \begin{pmatrix} I & -\boxed{A^{-1}}B \\ 0 & I \end{pmatrix} \begin{pmatrix} \boxed{A^{-1}} & 0 \\ 0 & \boxed{SC(L,T)^+} \end{pmatrix} \begin{pmatrix} I & 0 \\ -B^\top \boxed{A^{-1}} & I \end{pmatrix}$$

Schur Complement: $SC(L,T) = C - B^\top A^{-1} B$

Solving Laplacian System

$$L = \begin{pmatrix} I & 0 \\ B^\top A^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & SC(L,T) \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix}$$

$$\begin{array}{c} \rotatebox{90}{$\approx$} \\ L_1 \\ \vdots \\ L_d \end{array}$$

$$L \approx_C U_1^\top U_2^\top \cdots U_d^\top M U_d \cdots U_2 U_1$$

Solving Laplacian System

$$L = \begin{pmatrix} I & 0 \\ B^\top A^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & SC(L,T) \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix}$$

$$L \approx_C U_1^\top U_2^\top \cdots U_d^\top M U_d \cdots U_2 U_1$$

Apply Iterative Refinement for $\log\frac{1}{\delta}$ iterations

1. How to find an A_i that is easily invertible?
2. How to build a sparse approximation to the Schur complement implicitly?

Strongly Diagonally Dominant

A matrix A is said to be 5-Diagonally Dominant (DD) if

$$A_{ii} \geq 5 \sum_{j:j \neq i} |A_{ij}|$$

For a 5-DD matrix A, if D is its diagonal, $D \approx_{0.5} A$

5-DD matrices can be δ-approximately solved in $O(\log\frac{1}{\delta})$ iterations using iterative refinement

Finding 5-DD blocks

A is said to be 5-DD if $A_{ii} \geq 5 \sum_{j:j\neq i} |A_{ij}|$

[LPS '15, KLPSS '16] *Can find a 5-DD subblock of size n/40 in $O(m)$ time*

1. Pick each vertex to be in A independently with prob. 1/20
2. For vertex $i \in A$, $\mathbb{E}\left[\sum_{j\in S:j\neq i} | A_{ij} |\right] = \frac{A_{ii}}{20}$
3. By Markov, we have vertex $i \in A$ satisfies $\sum_{j:j\neq i} |A_{ij}| > \frac{A_{ii}}{5}$ with prob ¼
4. By another Markov, with probability 1/2, at least half of A set gives 5-DD subset

Solving Laplacian System

$$L = \begin{pmatrix} I & 0 \\ B^\top A^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & SC(L,T) \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix}$$

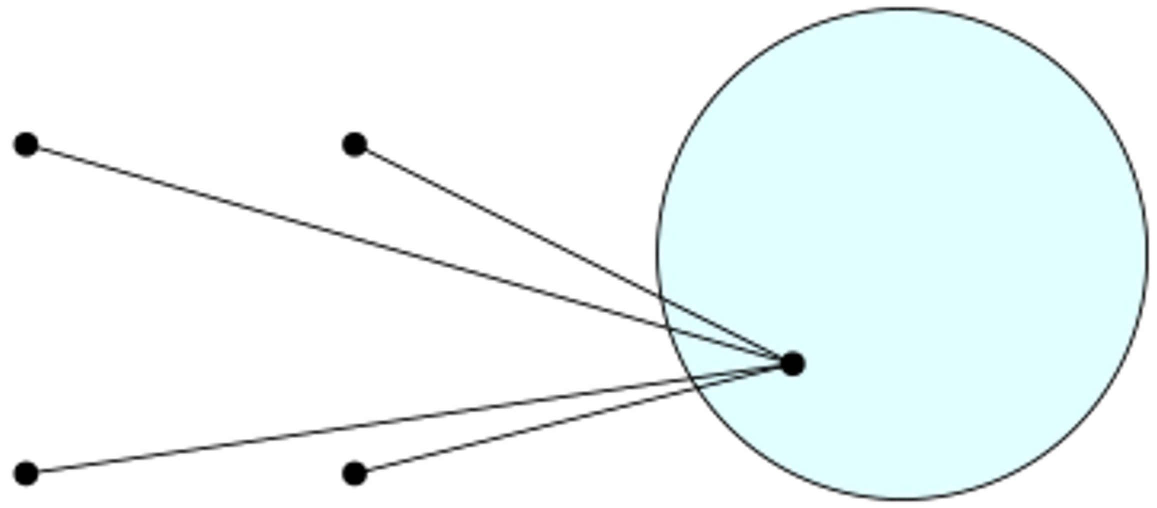
$$L \approx_C U_1^\top U_2^\top \cdots U_d^\top M U_d \cdots U_2 U_1$$

Apply Iterative Refinement for $O\left(\log\frac{1}{\delta}\right)$ iterations

1. ✓ How to find an A_i that is easily invertible?
2. How to build a sparse approximation to the Schur complement implicitly?

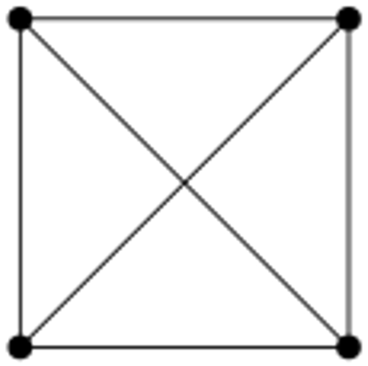
Challenge with (Block) Cholesky Factorization

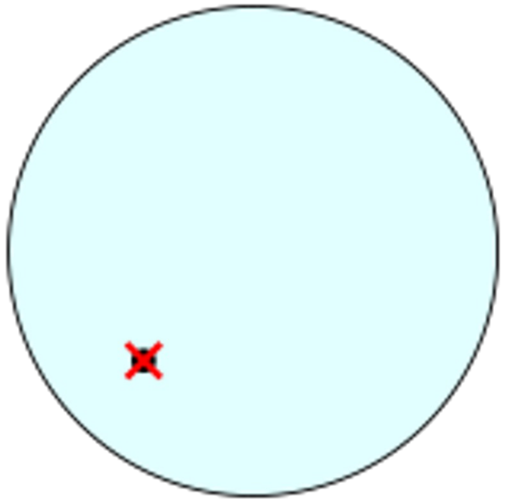
Fill-in phenomenon



Challenge with (Block) Cholesky Factorization

Fill-in phenomenon





Schur Complement = Random walks

Recall
$$L = \begin{array}{cc} & \begin{array}{cc} F & T \end{array} \\ & \begin{pmatrix} A & B \\ B^\top & C \end{pmatrix} \begin{array}{c} F \\ T \end{array} \end{array} \qquad SC(L,T) = C - B^\top A^{-1} B$$

Writing $A = D - H$, where D is a diagonal, and H has empty diagonal

$$SC(L,T) = C - B^\top \sum_i D^{-1}(HD^{-1})^i \, B$$

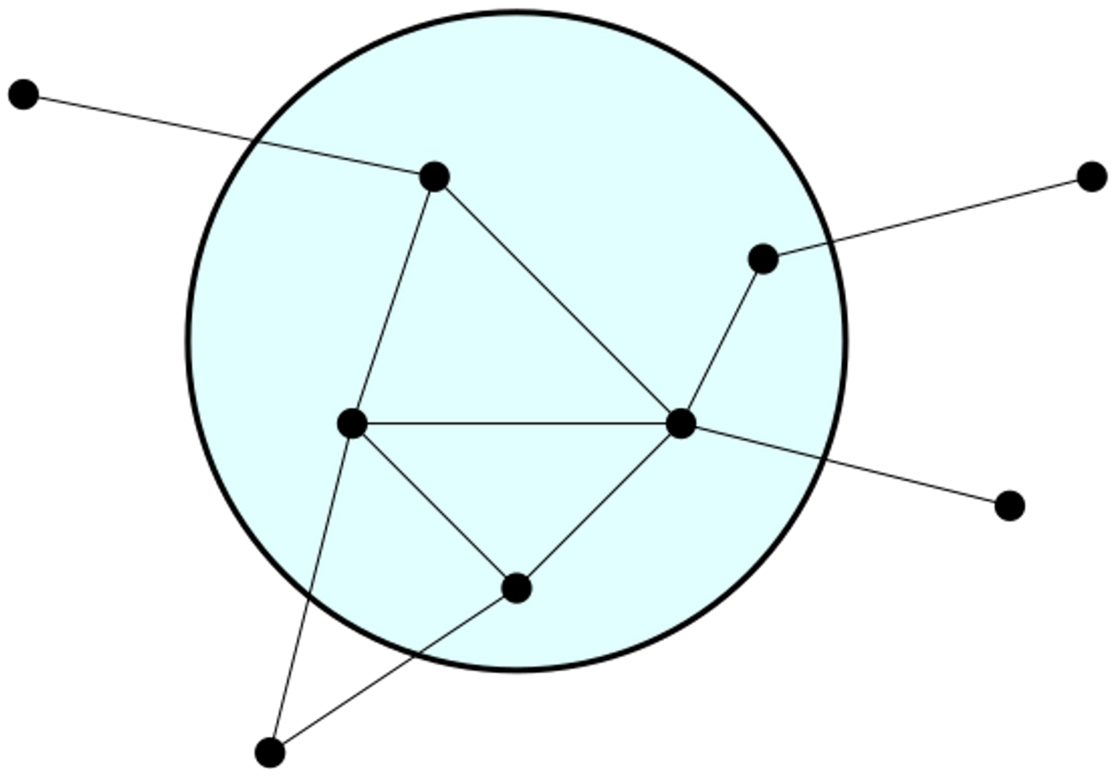
Recipe to build Schur Complement

1. Take each walk w starting at $x \in T$, walks over vertices in F, and ends at $y \in T$
2. Add (x,y) with weight $\frac{\prod weight\ of\ all\ edges\ in\ w}{\prod degree\ of\ all\ vertices\ from\ F\ in\ w}$

Schur Complement Sampling

Approach: Random Walk Sampling

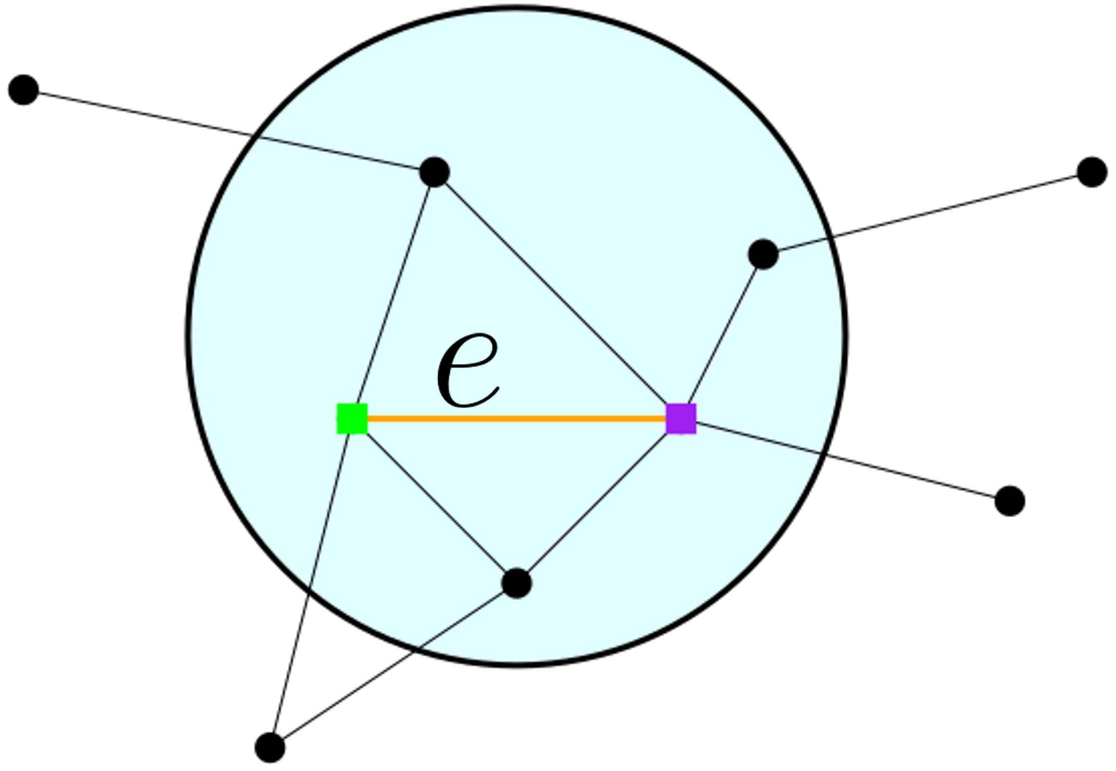
Unbiased *approximation of Schur Complement using "Terminal Random Walk"*



Schur Complement Sampling

Approach: Random Walk Sampling

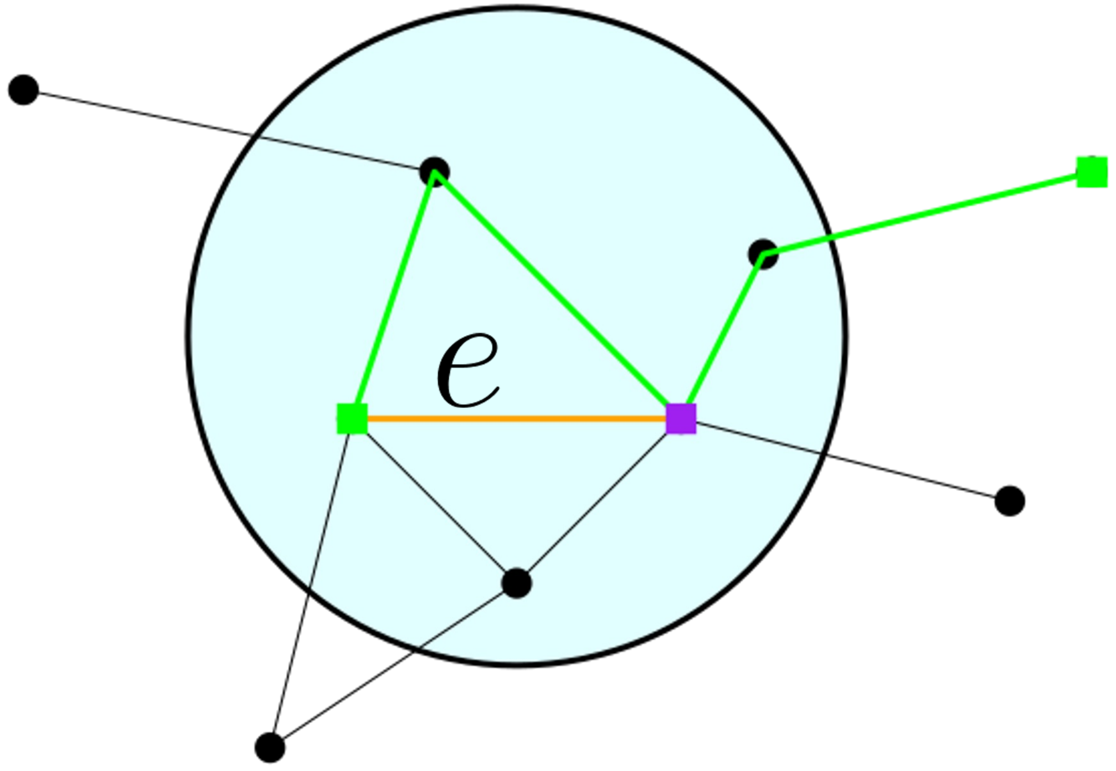
Unbiased *approximation of Schur Complement using "Terminal Random Walk"*



Schur Complement Sampling

Approach: Random Walk Sampling

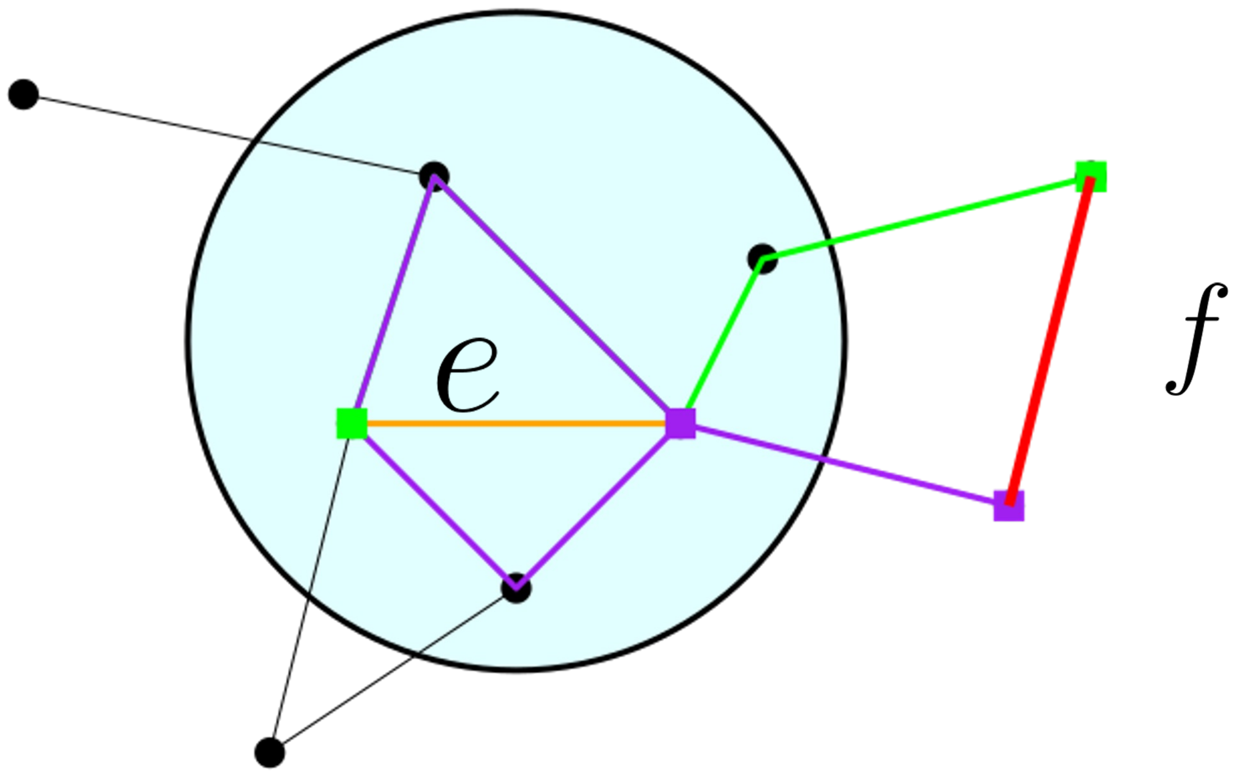
***Unbiased** approximation of Schur Complement using "Terminal Random Walk"*



Schur Complement Sampling

Approach: Random Walk Sampling

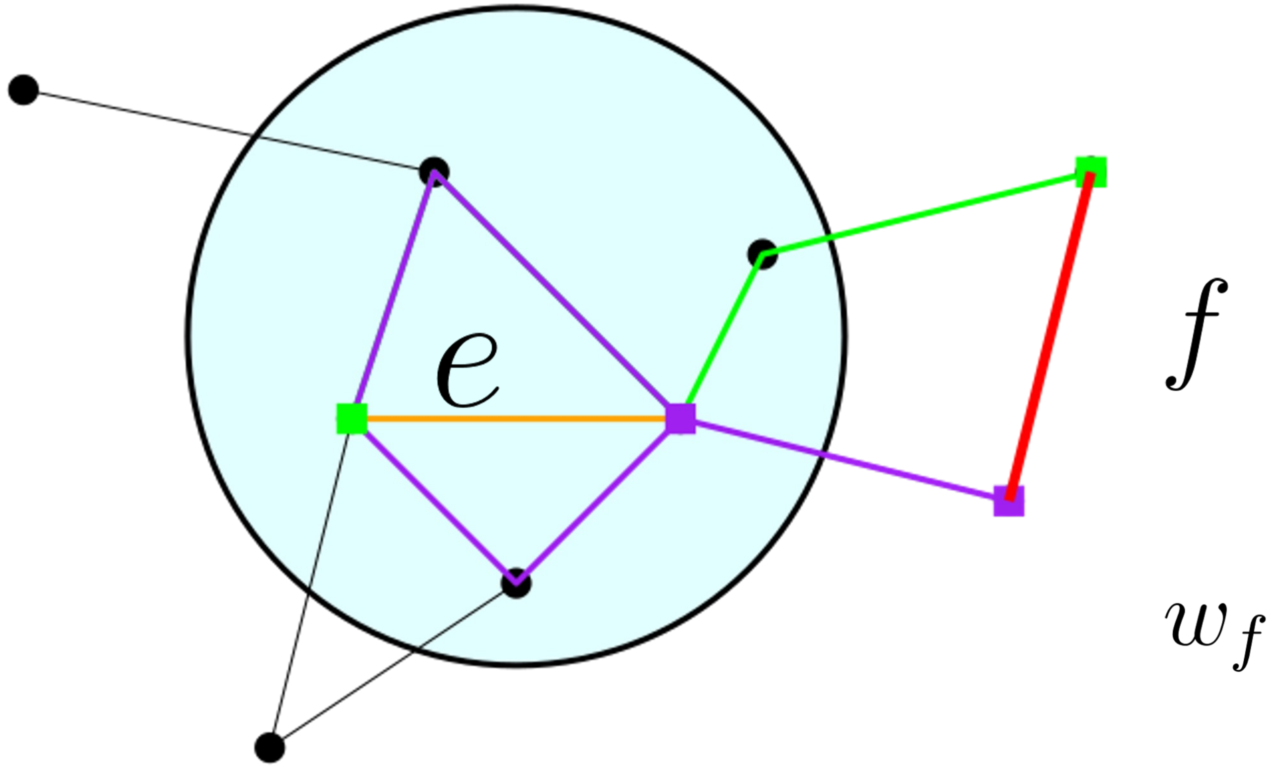
***Unbiased** approximation of Schur Complement using "Terminal Random Walk"*



Schur Complement Sampling

Approach: Random Walk Sampling

Unbiased *approximation of Schur Complement using "Terminal Random Walk"*



$$w_f = \frac{1}{\sum_{l \in W(e)} 1/w_l}$$

Schur Complement Sampling

Approach: Random Walk Sampling

Unbiased *approximation of Schur Complement using "Terminal Random Walk"*

1. For every edge (u, v),
   Random walk from u until it hits first vertex $x \in T$.
   Similarly walk from v until $y \in T$.
   Let w be the complete x to y walk.

2. Add edge (x, y) with weight $\frac{1}{\sum_{e \in w} \frac{1}{w_e}}$

Schur Complement Sampling

Key properties:

1. Each walk $O(\log m)$ w.h.p.
2. Total running time $O(m)$ w.h.p.
3. Total number of edges remains the same
4. Exercise: In expectation, you exactly get the Schur complement

Achieving Schur approximation

For ε-approximation,

need each sample to have leverage score $w_e R_G(e) \leq \frac{\varepsilon^2}{\log n}$

Key observation: If all original edges have leverage score at most $\frac{\varepsilon^2}{\log n}$,

then each sampled edge has leverage score at most $\frac{\varepsilon^2}{\log n}$

Achieving Schur approximation

Key observation: If all original edges have leverage score at most $\frac{\varepsilon^2}{\log n}$, then each sampled edge has leverage score at most $\frac{\varepsilon^2}{\log n}$

Using triangle inequality,

$$\frac{R(x,y)}{\frac{1}{w_1}+\frac{1}{w_2}+\frac{1}{w_3}} \le \frac{R(x,u)+R(u,v)+R(v,y)}{\frac{1}{w_1}+\frac{1}{w_2}+\frac{1}{w_3}}$$

$$= \frac{\frac{1}{w_1}\cdot w_1 R(x,u)+\frac{1}{w_2}\cdot w_2 R(u,v)+\frac{1}{w_3}\cdot w_3 R(v,y)}{\frac{1}{w_1}+\frac{1}{w_2}+\frac{1}{w_3}} \le \frac{\varepsilon^2}{\log n}$$

Achieve Schur approximation

1. Sampling with right expectation
2. Each sample has small leverage score $\frac{\varepsilon^2}{\log n}$

Matrix concentration guarantees ε-approx Schur complement

Achieve small leverage score by splitting edges beforehand

Recurse $\mathrm{O}(\log n)$ times with smaller epsilon for complete algorithm

Conclusion

A very simple Laplacian solver - $m \log^4 \text{n}$ time

Can be parallelized with $\log^2 n$ depth

Work can be improved to $m \log n + n \log^6 n$ by edge subsampling

Gives improved parallel algorithms for schur approximation, graph sparsification, effective resistance estimation

Thanks!

Laplacians

Symmetric $n \times n$ matrix,
associated with a weighted, undirected multi-graph $G = (V, E, w)$

$$n = |V|, \qquad m = |E|, \qquad w: E \to \mathbb{R}_+$$

Laplacian of a single unweighted edge (u, v)

Quadratic form

$$x^\top L x = (x_u - x_v)^2$$

Laplacians

Symmetric $n \times n$ matrix,
associated with a weighted, undirected multi-graph $G = (V, E, w)$

$$n = |V|, \qquad m = |E|, \qquad w: E \to \mathbb{R}_+$$

Laplacian of a single unweighted edge (u, v)

Quadratic form

$$x^\top L x = (x_u - x_v)^2$$

$$L = \begin{array}{cc} \begin{array}{ccc} u & & v \end{array} & \\ \begin{pmatrix} 1 & \cdots & -1 \\ \vdots & \ddots & \vdots \\ -1 & \cdots & 1 \end{pmatrix} & \begin{array}{c} u \\ \\ v \end{array} \end{array}$$

Laplacians

Symmetric $n{\times}n$ matrix,
associated with a weighted, undirected multi-graph $G = (V, E, w)$

$$n = |V|\,, \qquad m = |E|, \qquad w\colon E \to \mathbb{R}_+$$

Laplacian of G

Quadratic form

$$x^\top L x = \sum_{(u,v)\in E} w_{uv}(x_u - x_v)^2$$

Laplacians

Symmetric $n \times n$ matrix,
associated with a weighted, undirected multi-graph $G = (V, E, w)$

$$n = |V|, \qquad m = |E|, \qquad w: E \to \mathbb{R}_+$$

Laplacian of G

Quadratic form

$$x^\top L x = \sum_{(u,v)\in E} w_{uv}(x_u - x_v)^2$$

$$L = \sum_{(u,v)\in E} w_{uv} \begin{array}{cc} \begin{array}{ccc} u & & v \end{array} & \\ \begin{pmatrix} 1 & \cdots & -1 \\ \vdots & \ddots & \vdots \\ -1 & \cdots & 1 \end{pmatrix} & \begin{array}{c} u \\ \\ v \end{array} \end{array}$$