The Distributed Laplacian Paradigm

Algebraic Methods for Combinatorial Problems

Tijn de Vos

Part 1:
Algebra – it's not so bad

Part 2:
Computing Max Flow – it's not so easy

Please ask questions!

Q: What does Linear Algebra have to do with Graphs?

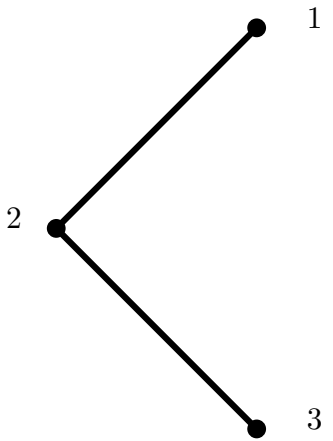
Q: What does Linear Algebra have to do with Graphs?

- $G = (V, E, w)$ with $|V| = n$, $|E| = m$, $w \colon E \to \mathbb{R}$
- Adjacency matrix: $\mathbf{A} \in \mathbb{R}^{n \times n}$ such that $\mathbf{A}_{ij} = w(ij)$

Q: What does Linear Algebra have to do with Graphs?

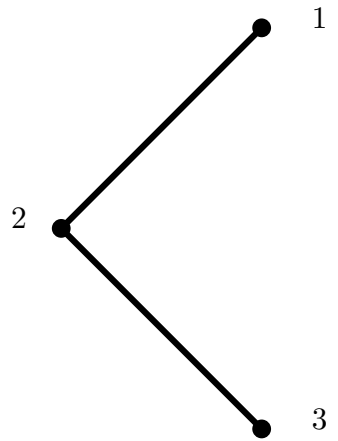
- $G = (V, E, w)$ with $|V| = n$, $|E| = m$, $w \colon E \to \mathbb{R}$
- Adjacency matrix: $\mathbf{A} \in \mathbb{R}^{n \times n}$ such that $\mathbf{A}_{ij} = w(ij)$
- E.g.

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$



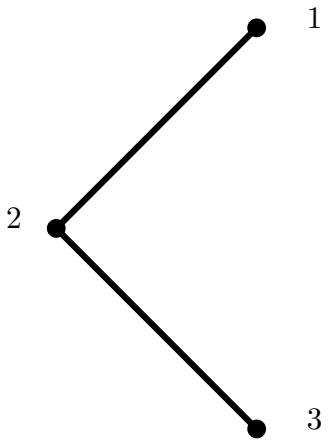
Q: What does Linear Algebra have to do with Graphs?

- $G = (V, E, w)$ with $|V| = n$, $|E| = m$, $w \colon E \to \mathbb{R}$
- Adjacency matrix: $\mathbf{A} \in \mathbb{R}^{n \times n}$ such that $\mathbf{A}_{ij} = w(ij)$
- E.g.
$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$
- $\mathbf{\Delta} \in \mathbb{R}^{n \times n}$ with degrees on the diagonal
- Laplacian matrix: $\mathbf{L} := \mathbf{\Delta} - \mathbf{A}$



Q: What does Linear Algebra have to do with Graphs?

- $G = (V, E, w)$ with $|V| = n$, $|E| = m$, $w \colon E \to \mathbb{R}$
- Adjacency matrix: $\mathbf{A} \in \mathbb{R}^{n \times n}$ such that $\mathbf{A}_{ij} = w(ij)$
- E.g.
$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$
- $\mathbf{\Delta} \in \mathbb{R}^{n \times n}$ with degrees on the diagonal
- Laplacian matrix: $\mathbf{L} := \mathbf{\Delta} - \mathbf{A}$
- E.g.
$$\mathbf{L} = \mathbf{\Delta} - \mathbf{A} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}$$



Q: Why do we care about this matrix?

Q: Why do we care about this matrix?

- It encodes the entire graph

Q: Why do we care about this matrix?

- It encodes the entire graph
- E.g., all *cuts*:

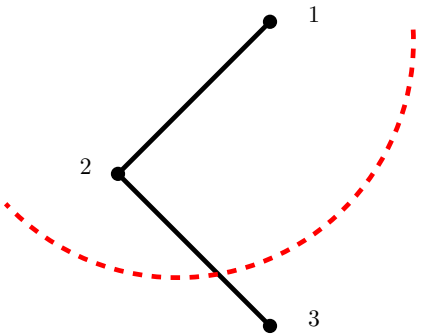
$$S \subseteq V \text{ then } |E(S, V \setminus S)| = \mathbf{1}_S^T \mathbf{L} \mathbf{1}_S$$

Q: Why do we care about this matrix?

- It encodes the entire graph
- E.g., all *cuts*:

$$S \subseteq V \text{ then } |E(S, V \setminus S)| = \mathbf{1}_S^T \mathbf{L} \mathbf{1}_S$$

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

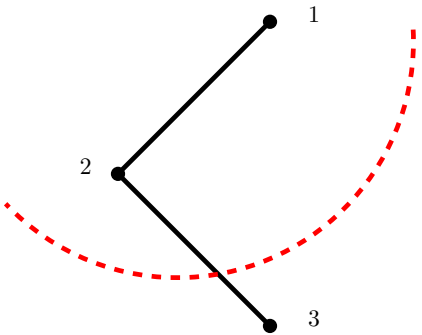


Q: Why do we care about this matrix?

- It encodes the entire graph
- E.g., all *cuts*:

$$S \subseteq V \text{ then } |E(S, V \setminus S)| = \mathbf{1}_S^T \mathbf{L} \mathbf{1}_S$$

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

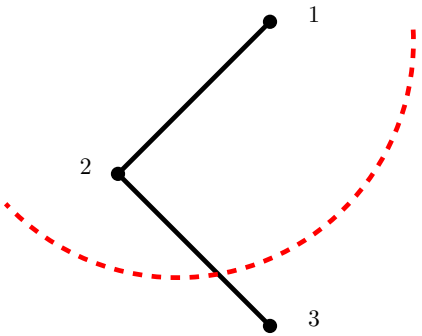


Q: Why do we care about this matrix?

- It encodes the entire graph
- E.g., all *cuts*:

$$S \subseteq V \text{ then } |E(S, V \setminus S)| = \mathbf{1}_S^T \mathbf{L} \mathbf{1}_S$$

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



- **L** can be dense and all cuts are a lot of vectors
- Instead: consider the eigenvalues

Q: What is an eigenvalue?

Q: What is an eigenvalue?

- Let $\mathbf{M} \in \mathbb{R}^{n \times n}$
- If $\mathbf{M}\mathbf{v} = \lambda \mathbf{v}$ for some $\lambda \in \mathbb{C}$, $\mathbf{v} \in \mathbb{R}^n$, then
  - λ is an eigenvalue
  - $\mathbf{v}$ is an eigenvector

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

Q: What is an eigenvalue?

- Let $\mathbf{M} \in \mathbb{R}^{n \times n}$
- If $\mathbf{Mv} = \lambda \mathbf{v}$ for some $\lambda \in \mathbb{C}$, $\mathbf{v} \in \mathbb{R}^n$, then
  - λ is an eigenvalue
  - $\mathbf{v}$ is an eigenvector

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

Q: What is an eigenvalue?

- Let $\mathbf{M} \in \mathbb{R}^{n \times n}$
- If $\mathbf{Mv} = \lambda \mathbf{v}$ for some $\lambda \in \mathbb{C}$, $\mathbf{v} \in \mathbb{R}^n$, then
  - λ is an eigenvalue
  - $\mathbf{v}$ is an eigenvector

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

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

Q: What is an eigenvalue?

- Let $\mathbf{M} \in \mathbb{R}^{n \times n}$
- If $\mathbf{Mv} = \lambda \mathbf{v}$ for some $\lambda \in \mathbb{C}$, $\mathbf{v} \in \mathbb{R}^n$, then
  - λ is an eigenvalue
  - $\mathbf{v}$ is an eigenvector

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

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

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ -6 \\ 3 \end{pmatrix} = 3 \cdot \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}$$

Q: What is an eigenvalue?

- Let $\mathbf{M} \in \mathbb{R}^{n \times n}$
- If $\mathbf{M}\mathbf{v} = \lambda \mathbf{v}$ for some $\lambda \in \mathbb{C}$, $\mathbf{v} \in \mathbb{R}^n$, then
  - λ is an eigenvalue
  - $\mathbf{v}$ is an eigenvector
- $\mathbf{M}$ has n eigenvalues

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

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

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ -6 \\ 3 \end{pmatrix} = 3 \cdot \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}$$

Q: What is an eigenvalue?

- Let $\mathbf{M} \in \mathbb{R}^{n \times n}$
- If $\mathbf{M}\mathbf{v} = \lambda\mathbf{v}$ for some $\lambda \in \mathbb{C}$, $\mathbf{v} \in \mathbb{R}^n$, then
  - λ is an eigenvalue
  - $\mathbf{v}$ is an eigenvector
- $\mathbf{M}$ has n eigenvalues
- Can assume $\mathbf{v}^T\mathbf{v} = \langle \mathbf{v}, \mathbf{v} \rangle = ||\mathbf{v}||^2 = 1$.
  - Since $\mathbf{M}(c\mathbf{v}) = c\mathbf{M}\mathbf{v} = c(\lambda\mathbf{v}) = \lambda(c\mathbf{v})$

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

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

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ -6 \\ 3 \end{pmatrix} = 3 \cdot \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}$$

Q: What is an eigenvalue?

- Let $\mathbf{M} \in \mathbb{R}^{n \times n}$
- If $\mathbf{M}\mathbf{v} = \lambda\mathbf{v}$ for some $\lambda \in \mathbb{C}$, $\mathbf{v} \in \mathbb{R}^n$, then
  - λ is an eigenvalue
  - $\mathbf{v}$ is an eigenvector
- $\mathbf{M}$ has n eigenvalues
- Can assume $\mathbf{v}^T\mathbf{v} = \langle \mathbf{v}, \mathbf{v} \rangle = ||\mathbf{v}||^2 = 1$.
  - Since $\mathbf{M}(c\mathbf{v}) = c\mathbf{M}\mathbf{v} = c(\lambda\mathbf{v}) = \lambda(c\mathbf{v})$
- If $\mathbf{M}$ is symmetric, then
  - All eigenvalues are real: $\lambda \in \mathbb{R}$
  - Eigenvectors are orthogonal

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

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

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ -6 \\ 3 \end{pmatrix} = 3 \cdot \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}$$

Q: What is an eigenvalue?

- Let $\mathbf{M} \in \mathbb{R}^{n \times n}$
- If $\mathbf{M}\mathbf{v} = \lambda\mathbf{v}$ for some $\lambda \in \mathbb{C}$, $\mathbf{v} \in \mathbb{R}^n$, then
  - λ is an eigenvalue
  - $\mathbf{v}$ is an eigenvector
- $\mathbf{M}$ has n eigenvalues
- Can assume $\mathbf{v}^T\mathbf{v} = \langle \mathbf{v}, \mathbf{v} \rangle = ||\mathbf{v}||^2 = 1$.
  - Since $\mathbf{M}(c\mathbf{v}) = c\mathbf{M}\mathbf{v} = c(\lambda\mathbf{v}) = \lambda(c\mathbf{v})$
- If $\mathbf{M}$ is symmetric, then
  - All eigenvalues are real: $\lambda \in \mathbb{R}$
  - Eigenvectors are orthogonal
- For Laplacians $\mathbf{L}$:
  - 0 is an eigenvalue
  - all eigenvalues are ≥ 0

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

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

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ -6 \\ 3 \end{pmatrix} = 3 \cdot \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}$$

Q: Why do we care about this matrix?

- The eigenvalues of **L** correspond to:

Q: Why do we care about this matrix?

- The eigenvalues of **L** correspond to:
  - The number of connected components
  - The size of independent sets (Hoffman's bound);
  - The chromatic number;
  - The average density of cuts;
  - The toughness of the graph;
  - The Hamiltonicity;
  - The matching number;
  - The existence of a perfect matching.

Q: Why do we care about this matrix?

- The eigenvalues of $\mathbf{L}$ correspond to:
  - The number of connected components
  - The size of independent sets (Hoffman's bound);
  - The chromatic number;
  - The average density of cuts;
  - The toughness of the graph;
  - The Hamiltonicity;
  - The matching number;
  - The existence of a perfect matching.
- Normalized Laplacian: $\mathbf{L} = \mathbf{I} - \mathbf{\Delta}^{-1/2}\mathbf{A}\mathbf{\Delta}^{-1/2}$

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & -1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1 & -1/\sqrt{2} \\ 0 & -1/\sqrt{2} & 1 \end{pmatrix}$$

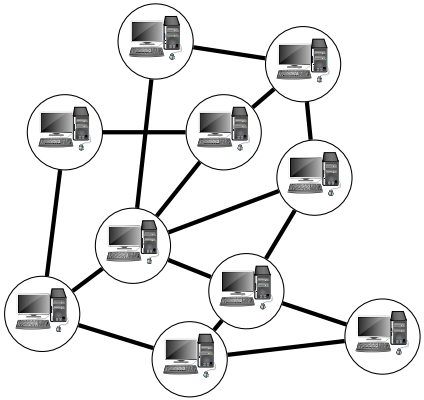
Q: Why do we care about this matrix?

- The eigenvalues of $\mathbf{L}$ correspond to:
  - The number of connected components
  - The size of independent sets (Hoffman's bound);
  - The chromatic number;
  - The average density of cuts;
  - The toughness of the graph;
  - The Hamiltonicity;
  - The matching number;
  - The existence of a perfect matching.
- Normalized Laplacian: $\mathbf{L} = \mathbf{I} - \mathbf{\Delta}^{-1/2}\mathbf{A}\mathbf{\Delta}^{-1/2}$
  - Sparsest cut = conductance = expansion
  - Mixing time
  - Bipartiteness

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & -1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1 & -1/\sqrt{2} \\ 0 & -1/\sqrt{2} & 1 \end{pmatrix}$$

Q: That sounds so sequential!

Intermezzo – The CONGEST Model



- $G = (V, E)$, $|V| = n$, $|E| = m$
- Communication over edges in synchronous rounds.
- Bandwidth $O(\log n)$ bits per edge.
- Broadcast CONGEST: Broadcast *same* message to all neighbors.

Q: That sounds so sequential!

Distributed Matrix-Vector Multiplication: 1 round

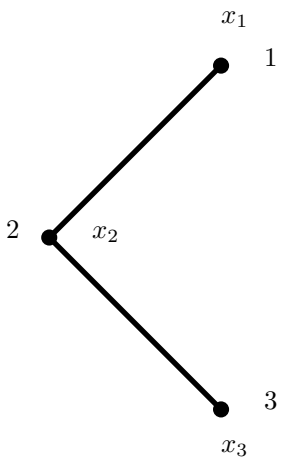
$$\mathbf{Lx} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Q: That sounds so sequential!

Distributed Matrix-Vector Multiplication: 1 round

$$\mathbf{Lx} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

- Input: node i knows x_i



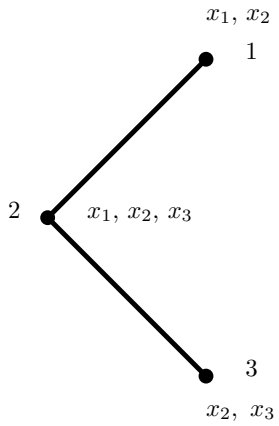
Q: That sounds so sequential!

Distributed Matrix-Vector Multiplication: 1 round

$$\mathbf{Lx} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

- Input: node i knows x_i
- Send x_i to your neighbors
- Compute internally $(\mathbf{Lx})_i$

$$(\mathbf{Lx})_1 = \begin{pmatrix} 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = x_1 - x_2$$



Q: That sounds so sequential!

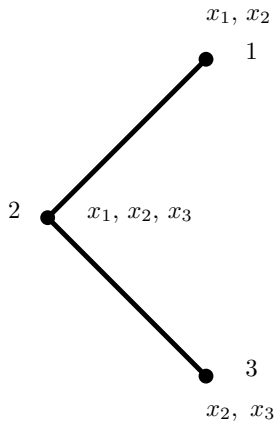
Distributed Matrix-Vector Multiplication: 1 round

$$\mathbf{Lx} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

- Input: node i knows x_i
- Send x_i to your neighbors
- Compute internally $(\mathbf{Lx})_i$

$$(\mathbf{Lx})_1 = \begin{pmatrix} 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = x_1 - x_2$$

- Output: node i knows $(\mathbf{Lx})_i$



Q: That sounds so sequential!

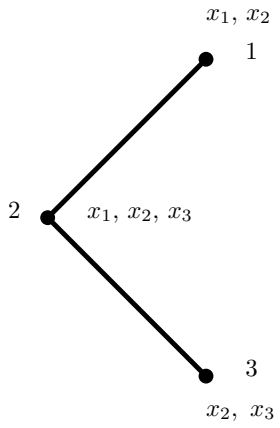
Distributed Matrix-Vector Multiplication: 1 round

$$\mathbf{Lx} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

- Input: node i knows x_i
- Send x_i to your neighbors
- Compute internally $(\mathbf{Lx})_i$

$$(\mathbf{Lx})_1 = \begin{pmatrix} 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = x_1 - x_2$$

- Output: node i knows $(\mathbf{Lx})_i$
- Enough for eigenvalue estimation [Maus, **dV** DISC'25]



Recap

- Can phrase the input as linear algebra ($\mathbf{L} = \mathbf{\Delta} - \mathbf{A}$)
- Can do linear operations
- Algebraic properties (eigenvalues) translate to graph properties

Q: Can we do anything more interesting?

Q: Can we do anything more interesting?

- Maximum Independent Set as a Linear Program (LP)

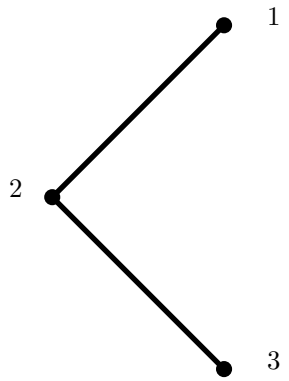
Q: Can we do anything more interesting?

- Maximum Independent Set as a Linear Program (LP)
- $\max_{\mathbf{x} \in \mathbb{R}^n} \sum_{v \in V} x_v$
  - $x_u + x_v \leq 1$ for all $uv \in E$
  - $x_v \in \{0, 1\}$ for all $v \in V$

Q: Can we do anything more interesting?

- Maximum Independent Set as a Linear Program (LP)
- $\max_{\mathbf{x} \in \mathbb{R}^n} \sum_{v \in V} x_v$
  - $x_u + x_v \leq 1$ for all $uv \in E$
  - $x_v \in \{0, 1\}$ for all $v \in V$
- Locally defined constraints:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

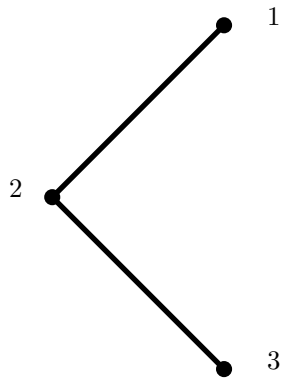


Q: Can we do anything more interesting?

- Maximum Independent Set as a Linear Program (LP)
- $\max_{\mathbf{x} \in \mathbb{R}^n} \sum_{v \in V} x_v$
  - $x_u + x_v \leq 1$ for all $uv \in E$
  - $x_v \in \{0, 1\}$ for all $v \in V$
- Locally defined constraints:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

- More generally: $\max_{x \in \mathbb{R}^n} \mathbf{c}^T \mathbf{x}$
  - $\mathbf{M}\mathbf{x} \leq \mathbf{b}$

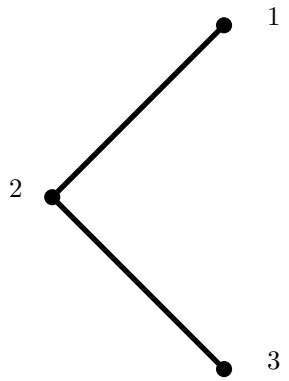


Q: Can we do anything more interesting?

- Maximum Independent Set as a Linear Program (LP)
- $\max_{\mathbf{x} \in \mathbb{R}^n} \sum_{v \in V} x_v$
  - $x_u + x_v \leq 1$ for all $uv \in E$
  - $x_v \in \{0, 1\}$ for all $v \in V$
- Locally defined constraints:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

- More generally: $\max_{x \in \mathbb{R}^n} \mathbf{c}^T \mathbf{x}$
  - $\mathbf{M}\mathbf{x} \leq \mathbf{b}$
  - Non-zero entries of $\mathbf{M}$ correspond to edges



Q: Are there also problems where algebraic methods are “necessary”?

Q: Are there also problems where algebraic methods are "necessary"?

In CONGEST:

- Approximate flow [Ghaffari, Karrenbauer, Kuhn, Lenzen, Patt-Shamir '15]
- Transshipment and shortest paths [Becker, Forster, Karrenbauer, Lenzen '17, Rozhoň (r) Grunau (r) Haeupler (r) Zuzic (r) Li '22, Zuzic (r) Goranci (r) Ye (r) Haeupler (r) Sun '22, Zuzic '23]
- Exact flow [Forster, Goranci, Liu, Peng, Sun, Ye '21, **dV** '23]
- Approximate Packing and Covering Linear Programs [Kuhn, Moscibroda, and Wattenhofer '06]

In LOCAL:

- Approximate Integer Packing and Covering Linear Programs [Ghaffari, Kuhn, and Maus '17, Chang and Li '23]

Q: Are there also problems where algebraic methods are "necessary"?

In CONGEST:

- Approximate flow [Ghaffari, Karrenbauer, Kuhn, Lenzen, Patt-Shamir '15]
- Transshipment and shortest paths [Becker, Forster, Karrenbauer, Lenzen '17, Rozhoň (r) Grunau (r) Haeupler (r) Zuzic (r) Li '22, Zuzic (r) Goranci (r) Ye (r) Haeupler (r) Sun '22, Zuzic '23]
  - Solved with Gradient Descent and Multiplicative Weight Update method
- Exact flow [Forster, Goranci, Liu, Peng, Sun, Ye '21, **dV** '23]
  - Solved with Interior Point Methods
- Approximate Packing and Covering Linear Programs [Kuhn, Moscibroda, and Wattenhofer '06]
  - Solved with a Primal-Dual approach

In LOCAL:

- Approximate Integer Packing and Covering Linear Programs [Ghaffari, Kuhn, and Maus '17, Chang and Li '23]
  - Solved with decomposition + exact local solution

Q: So what do I need the Laplacian for?

Q: So what do I need the Laplacian for?

- Constraint matrix in terms of Laplacian
  - Need to compute **Lx**

Q: So what do I need the Laplacian for?

- Constraint matrix in terms of Laplacian
  - Need to compute $\mathbf{Lx}$ (many methods)
  - Need to compute $\mathbf{L}^{-1}\mathbf{y}$ (some methods)

Q: So what do I need the Laplacian for?

- Constraint matrix in terms of Laplacian
  - Need to compute $\mathbf{Lx}$ (many methods)
  - Need to compute $\mathbf{L}^{-1}\mathbf{y}$ (some methods)

Laplacian Solver

- Given $\mathbf{y} \in \mathbb{R}^n$, let $\mathbf{x}^* \in \mathbb{R}^n$ s.t. $\mathbf{Lx}^* = \mathbf{y}$.
- Find $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{x} = \mathbf{x}^* \pm \varepsilon \mathbf{x}^*$.

Q: So what do I need the Laplacian for?

- Constraint matrix in terms of Laplacian
  - Need to compute $\mathbf{Lx}$ (many methods)
  - Need to compute $\mathbf{L}^{-1}\mathbf{y}$ (some methods)

Laplacian Solver

- Given $\mathbf{y} \in \mathbb{R}^n$, let $\mathbf{x}^* \in \mathbb{R}^n$ s.t. $\mathbf{Lx}^* = \mathbf{y}$.
- Find $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{x} = \mathbf{x}^* \pm \varepsilon \mathbf{x}^*$.

- Abuse of notation! Should write $x = x^* + z$ s.t. $||z|| \leq \varepsilon ||x^*||$

Q: So what do I need the Laplacian for?

- Constraint matrix in terms of Laplacian
  - Need to compute $\mathbf{Lx}$ (many methods)
  - Need to compute $\mathbf{L}^{-1}\mathbf{y}$ (some methods)

Laplacian Solver

- Given $\mathbf{y} \in \mathbb{R}^n$, let $\mathbf{x}^* \in \mathbb{R}^n$ s.t. $\mathbf{Lx}^* = \mathbf{y}$.
- Find $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{x} = \mathbf{x}^* \pm \varepsilon \mathbf{x}^*$.

- Abuse of notation! Should write $x = x^* + z$ s.t. $||z|| \leq \varepsilon ||x^*||$
- Generally can afford/need $\varepsilon = 1/\operatorname{poly} n$

Recap

- Can phrase graph problems as linear programs
- To solve a linear program we need
  - $\mathbf{Lx}$
  - $\mathbf{L}^{-1}\mathbf{y}$ (Laplacian Solving)

Laplacian Solving

- $\tilde{O}(m)$ sequential running time [Spielman, Teng '04]
- $\tilde{O}(m)$ work, polylogarithmic depth in PRAM [Peng, Spielman '14]

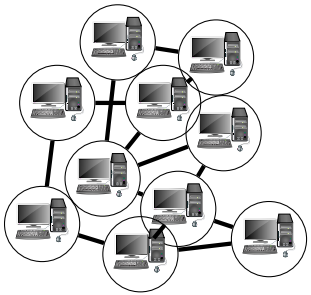
Laplacian Solving

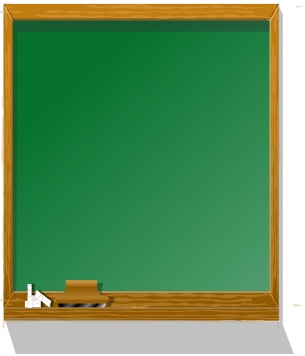
- $\tilde{O}(m)$ sequential running time [Spielman, Teng '04]
- $\tilde{O}(m)$ work, polylogarithmic depth in PRAM [Peng, Spielman '14]
- $\tilde{\Omega}(\sqrt{n} + D)$ rounds in CONGEST [Forster, Goranci, Liu, Peng, Sun, Ye '21]
- $n^{o(1)}(\sqrt{n} + D)$ rounds in CONGEST [Forster, Goranci, Liu, Peng, Sun, Ye '21]

Laplacian Solving

- $\tilde{O}(m)$ sequential running time [Spielman, Teng '04]
- $\tilde{O}(m)$ work, polylogarithmic depth in PRAM [Peng, Spielman '14]
- $\tilde{\Omega}(\sqrt{n} + D)$ rounds in CONGEST [Forster, Goranci, Liu, Peng, Sun, Ye '21]
- $n^{o(1)}(\sqrt{n} + D)$ rounds in CONGEST [Forster, Goranci, Liu, Peng, Sun, Ye '21]
- $O(\text{poly} \log n)$ rounds in Broadcast Congested Clique [Forster, **dV** '22]
- $n^{o(1)}$ rounds in Deterministic Congested Clique [Forster, **dV** '23]

Intermezzo – Congested Clique





- Nodes can communicate with all other nodes
- Synchronous, $O(\log n)$ message size.
- Analogues to sending/receiving n messages per node [Lenzen '13].
- Broadcast CC: Broadcast *same* message to all neighbors.

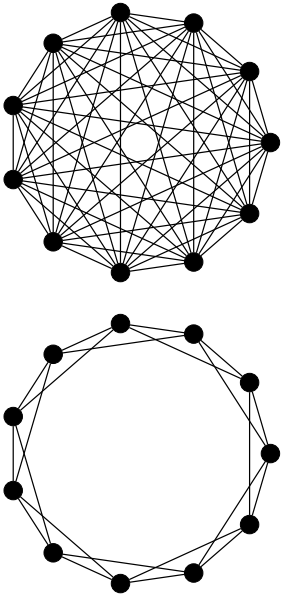
Q: How do we solve a Laplacian System?

- CONGEST algorithm is complicated
- Easy in CongestedClique
- Important tool: Spectral Sparsifiers

Q: What is a Spectral Sparsifier?

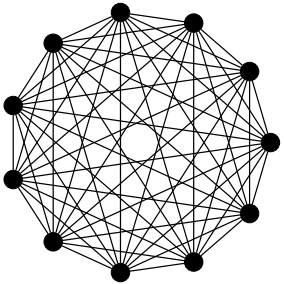
Q: What is a Spectral Sparsifier?

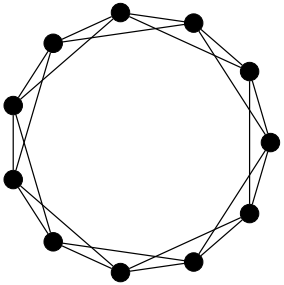
- Sparsifier $H \subseteq G$ is a (reweighted) sparse subgraph of G such that . . . of G is approximately maintained



Q: What is a Spectral Sparsifier?

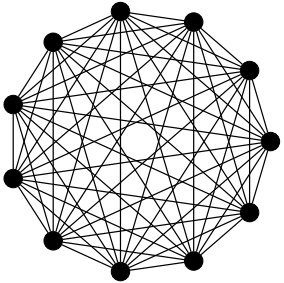
- Sparsifier $H \subseteq G$ is a (reweighted) sparse subgraph of G such that ... of G is approximately maintained
- A spectral sparsifier keeps spectral properties.

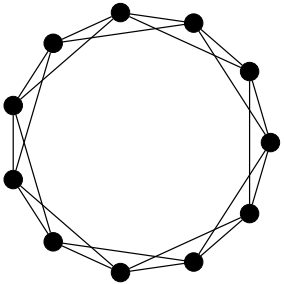




Q: What is a Spectral Sparsifier?

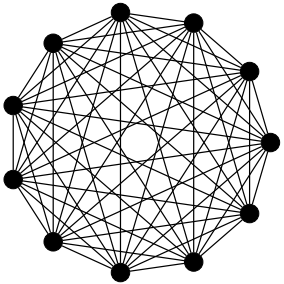
- Sparsifier $H \subseteq G$ is a (reweighted) sparse subgraph of G such that ... of G is approximately maintained
- A spectral sparsifier keeps spectral properties.
  - Eigenvalues of $\mathbf{L}_H \approx$ eigenvalues of $\mathbf{L}_G$

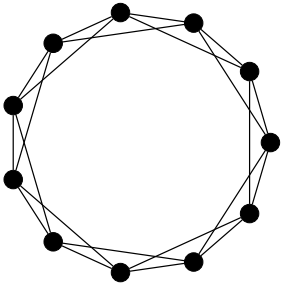




Q: What is a Spectral Sparsifier?

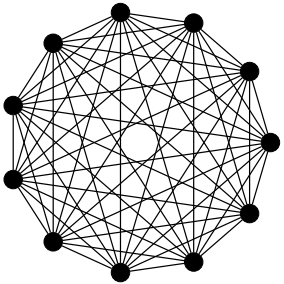
- Sparsifier $H \subseteq G$ is a (reweighted) sparse subgraph of G such that ... of G is approximately maintained
- A spectral sparsifier keeps spectral properties.
  - Eigenvalues of $\mathbf{L}_H \approx$ eigenvalues of $\mathbf{L}_G$
    - ★ Eigenvalue λ of G: $\mathbf{L}_G\mathbf{v} = \lambda\mathbf{v}$

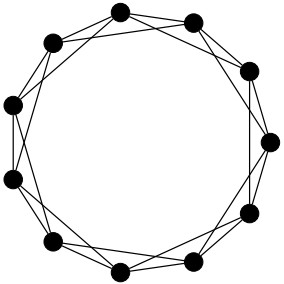




Q: What is a Spectral Sparsifier?

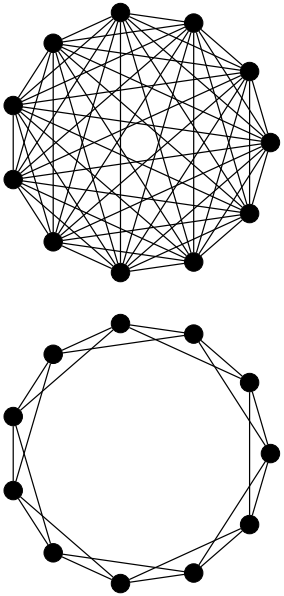
- Sparsifier $H \subseteq G$ is a (reweighted) sparse subgraph of G such that ... of G is approximately maintained
- A spectral sparsifier keeps spectral properties.
  - Eigenvalues of $\mathbf{L}_H \approx$ eigenvalues of $\mathbf{L}_G$
    - Eigenvalue λ of G: $\mathbf{L}_G\mathbf{v} = \lambda\mathbf{v}$
    - Can assume $\mathbf{v}^T\mathbf{v} = 1$, so $\lambda = \mathbf{v}^T\mathbf{L}_G\mathbf{v}$





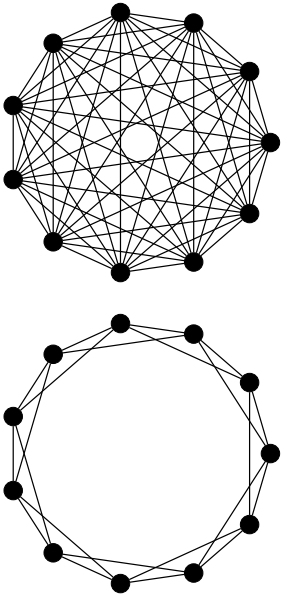
Q: What is a Spectral Sparsifier?

- Sparsifier $H \subseteq G$ is a (reweighted) sparse subgraph of G such that ... of G is approximately maintained
- A spectral sparsifier keeps spectral properties.
  - Eigenvalues of $\mathbf{L}_H \approx$ eigenvalues of $\mathbf{L}_G$
    - Eigenvalue λ of G: $\mathbf{L}_G\mathbf{v} = \lambda\mathbf{v}$
    - Can assume $\mathbf{v}^T\mathbf{v} = 1$, so $\lambda = \mathbf{v}^T\mathbf{L}_G\mathbf{v}$
  - A cut value in $H \approx$ a value cut in G



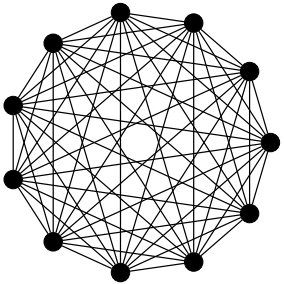
Q: What is a Spectral Sparsifier?

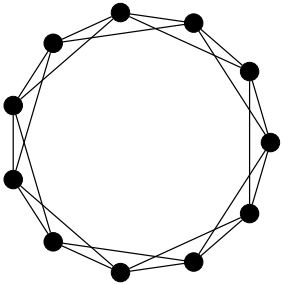
- Sparsifier $H \subseteq G$ is a (reweighted) sparse subgraph of G such that . . . of G is approximately maintained
- A spectral sparsifier keeps spectral properties.
  - Eigenvalues of $\mathbf{L}_H \approx$ eigenvalues of $\mathbf{L}_G$
    - Eigenvalue λ of G: $\mathbf{L}_G\mathbf{v} = \lambda\mathbf{v}$
    - Can assume $\mathbf{v}^T\mathbf{v} = 1$, so $\lambda = \mathbf{v}^T\mathbf{L}_G\mathbf{v}$
  - A cut value in $H \approx$ a value cut in G
    - Recall: $|E(S, V \setminus S)| = \mathbf{1}_S^T\mathbf{L}_G\mathbf{1}_S$



Q: What is a Spectral Sparsifier?

- Sparsifier $H \subseteq G$ is a (reweighted) sparse subgraph of G such that . . . of G is approximately maintained
- A spectral sparsifier keeps spectral properties.
  - Eigenvalues of $\mathbf{L}_H \approx$ eigenvalues of $\mathbf{L}_G$
    - Eigenvalue λ of G: $\mathbf{L}_G\mathbf{v} = \lambda\mathbf{v}$
    - Can assume $\mathbf{v}^T\mathbf{v} = 1$, so $\lambda = \mathbf{v}^T\mathbf{L}_G\mathbf{v}$
  - A cut value in $H \approx$ a value cut in G
    - Recall: $|E(S, V \setminus S)| = \mathbf{1}_S^T\mathbf{L}_G\mathbf{1}_S$
    - $\mathbf{x}^T\mathbf{L}_G\mathbf{x}$ for $\mathbf{x} \in \{0, 1\}^n$





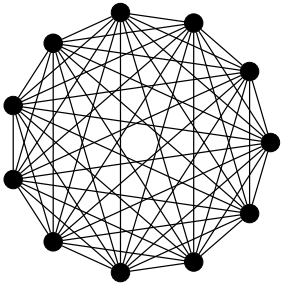
Q: What is a Spectral Sparsifier?

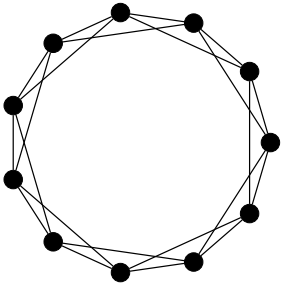
- Sparsifier $H \subseteq G$ is a (reweighted) sparse subgraph of G such that ... of G is approximately maintained
- A spectral sparsifier keeps spectral properties.
  - Eigenvalues of $\mathbf{L}_H \approx$ eigenvalues of $\mathbf{L}_G$
    - Eigenvalue λ of G: $\mathbf{L}_G\mathbf{v} = \lambda\mathbf{v}$
    - Can assume $\mathbf{v}^T\mathbf{v} = 1$, so $\lambda = \mathbf{v}^T\mathbf{L}_G\mathbf{v}$
  - A cut value in $H \approx$ a value cut in G
    - Recall: $|E(S, V \setminus S)| = \mathbf{1}_S^T\mathbf{L}_G\mathbf{1}_S$
    - $\mathbf{x}^T\mathbf{L}_G\mathbf{x}$ for $\mathbf{x} \in \{0,1\}^n$

Definition

$H \subseteq G$ is a $(1 \pm \varepsilon)$-spectral sparsifier of G if

$$(1-\varepsilon)\mathbf{x}^\mathsf{T}\mathbf{L}_H\mathbf{x} \leq \mathbf{x}^\mathsf{T}\mathbf{L}_G\mathbf{x} \leq (1+\varepsilon)\mathbf{x}^\mathsf{T}\mathbf{L}_H\mathbf{x} \qquad \forall \mathbf{x} \in \mathbb{R}^n.$$





Spectral Sparsifier in the Congested Clique

Spectral Sparsifier in the Congested Clique

Spectral Sparsifer in the CongestedClique

- $H \leftarrow 0$
- For $i = 1, \ldots, O(\log n)$:
  - For $j = 1, \ldots, O(\log n)$:
    - Compute an $O(\log n)$-spanner S of G
    - $H \leftarrow H \cup S$
    - $G \leftarrow G \setminus H$
  - Keep each edge in G with probability $\frac{1}{4}$.
- Return H

Laplacian Solving in the Congested Clique

- Compute a spectral sparsifier H of size $\tilde{O}(n)$

Laplacian Solving in the Congested Clique

- Compute a spectral sparsifier H of size $\tilde{O}(n)$
- Make H known to every node

Laplacian Solving in the Congested Clique

- Compute a spectral sparsifier H of size $\tilde{O}(n)$
- Make H known to every node
- Internally solve $\mathbf{L}_H \mathbf{x} = \mathbf{y}$

Laplacian Solving in the Congested Clique

- Compute a spectral sparsifier H of size $\tilde{O}(n)$
- Make H known to every node
- Internally solve $\mathbf{L}_H\mathbf{x} = \mathbf{y}$
- Output x

Accuracy

Preconditioned Chebyshev Iteration [Axelsson '93, Saad '03]

Even with a low-accuracy sparsifier, we can get high-accuracy solution to $\mathbf{L}_G\mathbf{x} = \mathbf{y}$ in $O(\log(1/\varepsilon))$ iterations.

Accuracy

Preconditioned Chebyshev Iteration [Axelsson '93, Saad '03]

Even with a low-accuracy sparsifier, we can get high-accuracy solution to $\mathbf{L}_G\mathbf{x} = \mathbf{y}$ in $O(\log(1/\varepsilon))$ iterations.

- Fix a δ-spectral sparsifier H of G (think $\delta = \frac{1}{2}$)

Accuracy

Preconditioned Chebyshev Iteration [Axelsson '93, Saad '03]

Even with a low-accuracy sparsifier, we can get high-accuracy solution to $\mathbf{L}_G\mathbf{x} = \mathbf{y}$ in $O(\log(1/\varepsilon))$ iterations.

- Fix a δ-spectral sparsifier H of G (think $\delta = \frac{1}{2}$)
- $\mathbf{x}_0 \leftarrow \mathbf{L}_H^{-1}\mathbf{y}$

Accuracy

Preconditioned Chebyshev Iteration [Axelsson '93, Saad '03]

Even with a low-accuracy sparsifier, we can get high-accuracy solution to $\mathbf{L}_G\mathbf{x} = \mathbf{y}$ in $O(\log(1/\varepsilon))$ iterations.

- Fix a δ-spectral sparsifier H of G (think $\delta = \frac{1}{2}$)
- $\mathbf{x}_0 \leftarrow \mathbf{L}_H^{-1}\mathbf{y}$
- Note $\mathbf{L}_G\mathbf{x}_0 = \mathbf{L}_G\mathbf{L}_H^{-1}\mathbf{y} = \mathbf{L}_G(\mathbf{L}_G^{-1}\mathbf{y} \pm \delta\mathbf{L}_G^{-1}\mathbf{y}) = \mathbf{y} \pm \delta\mathbf{y}$

Accuracy

Preconditioned Chebyshev Iteration [Axelsson '93, Saad '03]

Even with a low-accuracy sparsifier, we can get high-accuracy solution to $\mathbf{L}_G\mathbf{x} = \mathbf{y}$ in $O(\log(1/\varepsilon))$ iterations.

- Fix a δ-spectral sparsifier H of G (think $\delta = \frac{1}{2}$)
- $\mathbf{x}_0 \leftarrow \mathbf{L}_H^{-1}\mathbf{y}$
- Note $\mathbf{L}_G\mathbf{x}_0 = \mathbf{L}_G\mathbf{L}_H^{-1}\mathbf{y} = \mathbf{L}_G(\mathbf{L}_G^{-1}\mathbf{y} \pm \delta\mathbf{L}_G^{-1}\mathbf{y}) = \mathbf{y} \pm \delta\mathbf{y}$
- $\mathbf{x}_1 \leftarrow \mathbf{L}_H^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0)$

Accuracy

Preconditioned Chebyshev Iteration [Axelsson '93, Saad '03]

Even with a low-accuracy sparsifier, we can get high-accuracy solution to $\mathbf{L}_G\mathbf{x} = \mathbf{y}$ in $O(\log(1/\varepsilon))$ iterations.

- Fix a δ-spectral sparsifier H of G (think $\delta = \frac{1}{2}$)
- $\mathbf{x}_0 \leftarrow \mathbf{L}_H^{-1}\mathbf{y}$
- Note $\mathbf{L}_G\mathbf{x}_0 = \mathbf{L}_G\mathbf{L}_H^{-1}\mathbf{y} = \mathbf{L}_G(\mathbf{L}_G^{-1}\mathbf{y} \pm \delta\mathbf{L}_G^{-1}\mathbf{y}) = \mathbf{y} \pm \delta\mathbf{y}$
- $\mathbf{x}_1 \leftarrow \mathbf{L}_H^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0)$
- And
$$\mathbf{L}_G(\mathbf{x}_0 + \mathbf{x}_1) = \mathbf{L}_G\mathbf{x}_0 + \mathbf{L}_G\mathbf{L}_H^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0)$$

Accuracy

Preconditioned Chebyshev Iteration [Axelsson '93, Saad '03]

Even with a low-accuracy sparsifier, we can get high-accuracy solution to $\mathbf{L}_G\mathbf{x} = \mathbf{y}$ in $O(\log(1/\varepsilon))$ iterations.

- Fix a δ-spectral sparsifier H of G (think $\delta = \frac{1}{2}$)
- $\mathbf{x}_0 \leftarrow \mathbf{L}_H^{-1}\mathbf{y}$
- Note $\mathbf{L}_G\mathbf{x}_0 = \mathbf{L}_G\mathbf{L}_H^{-1}\mathbf{y} = \mathbf{L}_G(\mathbf{L}_G^{-1}\mathbf{y} \pm \delta\mathbf{L}_G^{-1}\mathbf{y}) = \mathbf{y} \pm \delta\mathbf{y}$
- $\mathbf{x}_1 \leftarrow \mathbf{L}_H^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0)$
- And

$$\begin{aligned}\mathbf{L}_G(\mathbf{x}_0 + \mathbf{x}_1) &= \mathbf{L}_G\mathbf{x}_0 + \mathbf{L}_G\mathbf{L}_H^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0) \\ &= \mathbf{L}_G\mathbf{x}_0 + \mathbf{L}_G(\mathbf{L}_G^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0) \pm \delta\mathbf{L}_G^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0))\end{aligned}$$

Accuracy

Preconditioned Chebyshev Iteration [Axelsson '93, Saad '03]

Even with a low-accuracy sparsifier, we can get high-accuracy solution to $\mathbf{L}_G\mathbf{x} = \mathbf{y}$ in $O(\log(1/\varepsilon))$ iterations.

- Fix a δ-spectral sparsifier H of G (think $\delta = \frac{1}{2}$)
- $\mathbf{x}_0 \leftarrow \mathbf{L}_H^{-1}\mathbf{y}$
- Note $\mathbf{L}_G\mathbf{x}_0 = \mathbf{L}_G\mathbf{L}_H^{-1}\mathbf{y} = \mathbf{L}_G(\mathbf{L}_G^{-1}\mathbf{y} \pm \delta\mathbf{L}_G^{-1}\mathbf{y}) = \mathbf{y} \pm \delta\mathbf{y}$
- $\mathbf{x}_1 \leftarrow \mathbf{L}_H^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0)$
- And

$$\begin{aligned}\mathbf{L}_G(\mathbf{x}_0 + \mathbf{x}_1) &= \mathbf{L}_G\mathbf{x}_0 + \mathbf{L}_G\mathbf{L}_H^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0) \\ &= \mathbf{L}_G\mathbf{x}_0 + \mathbf{L}_G(\mathbf{L}_G^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0) \pm \delta\mathbf{L}_G^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0)) \\ &= y \pm \delta(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0) = \mathbf{y} \pm \delta^2\mathbf{y}\end{aligned}$$

Accuracy

Preconditioned Chebyshev Iteration [Axelsson '93, Saad '03]

Even with a low-accuracy sparsifier, we can get high-accuracy solution to $\mathbf{L}_G\mathbf{x} = \mathbf{y}$ in $O(\log(1/\varepsilon))$ iterations.

- Fix a δ-spectral sparsifier H of G (think $\delta = \frac{1}{2}$)
- $\mathbf{x}_0 \leftarrow \mathbf{L}_H^{-1}\mathbf{y}$
- Note $\mathbf{L}_G\mathbf{x}_0 = \mathbf{L}_G\mathbf{L}_H^{-1}\mathbf{y} = \mathbf{L}_G(\mathbf{L}_G^{-1}\mathbf{y} \pm \delta\mathbf{L}_G^{-1}\mathbf{y}) = \mathbf{y} \pm \delta\mathbf{y}$
- $\mathbf{x}_1 \leftarrow \mathbf{L}_H^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0)$
- And

$$\begin{aligned}\mathbf{L}_G(\mathbf{x}_0 + \mathbf{x}_1) &= \mathbf{L}_G\mathbf{x}_0 + \mathbf{L}_G\mathbf{L}_H^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0) \\ &= \mathbf{L}_G\mathbf{x}_0 + \mathbf{L}_G(\mathbf{L}_G^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0) \pm \delta\mathbf{L}_G^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0)) \\ &= y \pm \delta(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0) = \mathbf{y} \pm \delta^2\mathbf{y}\end{aligned}$$

- Output $\mathbf{x} = \sum_{j<i} \mathbf{x}_j$

Accuracy

Preconditioned Chebyshev Iteration [Axelsson '93, Saad '03]

Even with a low-accuracy sparsifier, we can get high-accuracy solution to $\mathbf{L}_G\mathbf{x} = \mathbf{y}$ in $O(\log(1/\varepsilon))$ iterations.

- Fix a δ-spectral sparsifier H of G (think $\delta = \frac{1}{2}$)
- $\mathbf{x}_0 \leftarrow \mathbf{L}_H^{-1}\mathbf{y}$
- Note $\mathbf{L}_G\mathbf{x}_0 = \mathbf{L}_G\mathbf{L}_H^{-1}\mathbf{y} = \mathbf{L}_G(\mathbf{L}_G^{-1}\mathbf{y} \pm \delta\mathbf{L}_G^{-1}\mathbf{y}) = \mathbf{y} \pm \delta\mathbf{y}$
- $\mathbf{x}_1 \leftarrow \mathbf{L}_H^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0)$
- And

$$\begin{aligned}\mathbf{L}_G(\mathbf{x}_0 + \mathbf{x}_1) &= \mathbf{L}_G\mathbf{x}_0 + \mathbf{L}_G\mathbf{L}_H^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0) \\ &= \mathbf{L}_G\mathbf{x}_0 + \mathbf{L}_G(\mathbf{L}_G^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0) \pm \delta\mathbf{L}_G^{-1}(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0)) \\ &= y \pm \delta(\mathbf{y} - \mathbf{L}_G\mathbf{x}_0) = \mathbf{y} \pm \delta^2\mathbf{y}\end{aligned}$$

- Output $\mathbf{x} = \sum_{j<i}\mathbf{x}_j$
- $\mathbf{L}_G\mathbf{x} = \mathbf{y} \pm \delta^i\mathbf{y}$

Laplacian Solvers – Recap

- $\tilde{O}(m)$ sequential running time [Spielman, Teng '04]
- $\tilde{O}(m)$ work, $O(\text{poly}\log n)$ depth in PRAM [Peng, Spielman '14]
- $\tilde{\Omega}(\sqrt{n}+D)$ rounds in CONGEST [Forster, Goranci, Liu, Peng, Sun, Ye '21]
- $n^{o(1)}(\sqrt{n}+D)$ rounds in CONGEST [Forster, Goranci, Liu, Peng, Sun, Ye '21]
- $O(\text{poly}\log n)$ rounds in Broadcast Congested Clique [Forster, **dV** '22]
- $n^{o(1)}$ rounds in Deterministic Congested Clique [Forster, **dV** '23]

- In CongestedClique, we can compute Laplacian solving via spectral sparsifiers
- We can compute spectral sparsifiers via spanners
- We can boost the accuracy of a solver ($\log(1/\varepsilon)$ iterations)

Q: Is Laplacian solving always so expensive?

- Universal Optimality: the $\tilde{\Omega}(\sqrt{n} + D)$ lower bound is not for all graph topologies

Q: Is Laplacian solving always so expensive?

- Universal Optimality: the $\tilde{\Omega}(\sqrt{n} + D)$ lower bound is not for all graph topologies
- The Shortcut Quality $SQ(G)$ is a more precise lower bound: $D \leq SQ(G) \leq D + \sqrt{n}$

Q: Is Laplacian solving always so expensive?

- Universal Optimality: the $\tilde{\Omega}(\sqrt{n} + D)$ lower bound is not for all graph topologies
- The Shortcut Quality $SQ(G)$ is a more precise lower bound: $D \leq SQ(G) \leq D + \sqrt{n}$
- Topology dependent Laplacian solver [Anagnostides, Gouleakis, Lenzen '21, Anagnostides Ⓡ Lenzen Ⓡ Haeupler Ⓡ Zuzic Ⓡ Gouleakis '22]
  - Lower bound: $\tilde{\Omega}(SQ(G))$
  - Upper bound: $n^{o(1)}\,\mathrm{poly}(SQ(G))\log(1/\varepsilon)$ rounds
  - $n^{o(1)}D$ rounds in
    - ★ planar graphs
    - ★ expander graphs
    - ★ $n^{o(1)}$-genus graphs
    - ★ $n^{o(1)}$-treewidth graphs
    - ★ excluded-minor graphs

Part 1:
Algebra – it's not so bad

Part 2:
Computing Max Flow – it's not so easy

Flow

Definition

$G = (V, E)$ directed, capacities $c \in \mathbb{Z}^m_{\geq 0}$, costs $q \in \mathbb{Z}^m$, source and target $s, t \in V$. The minimum cost (maximum) flow problem is to find the $s - t$ flow $f \in \mathbb{R}^m$ of minimum cost, among all flows of maximum value.

- Generalizes Max Flow and Negative Weight Shortest Path.
- Min Cost Flow is a linear program.

Flow

Definition

$G = (V, E)$ directed, capacities $c \in \mathbb{Z}^m_{\geq 0}$, costs $q \in \mathbb{Z}^m$, source and target $s, t \in V$. The minimum cost (maximum) flow problem is to find the $s - t$ flow $f \in \mathbb{R}^m$ of minimum cost, among all flows of maximum value.

- Generalizes Max Flow and Negative Weight Shortest Path.
- Min Cost Flow is a linear program.

Theorem (Informal)

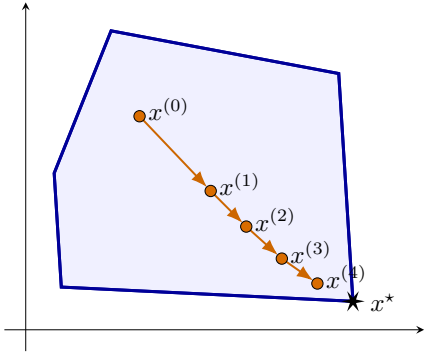
Min Cost Flow with $\tilde{O}(\sqrt{n})$ Laplacian solves in the CONGEST/BroadcastCongestedClique *model.*

Interior-Point Methods

Interior-Point Methods

- Goal: Solve a linear program

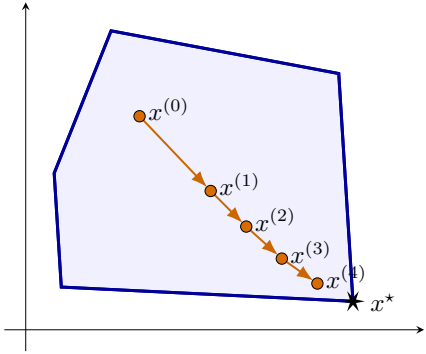
$$\min_{\mathbf{x}\in\mathbb{R}^n} \mathbf{c}^T\mathbf{x} \quad \text{s.t.} \quad \mathbf{M}\mathbf{x} = \mathbf{b}, \;\; \mathbf{x} \geq 0.$$



Interior-Point Methods

- Goal: Solve a linear program

$$\min_{\mathbf{x}\in\mathbb{R}^n} \mathbf{c}^T\mathbf{x} \quad \text{s.t.} \quad \mathbf{M}\mathbf{x} = \mathbf{b}, \;\; \mathbf{x} \geq 0.$$

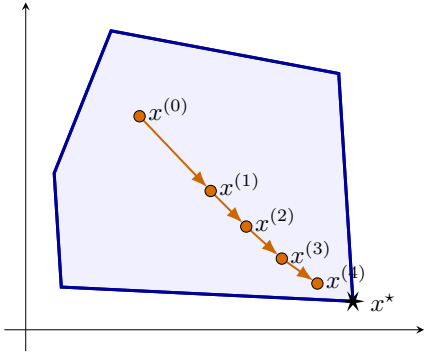


Interior-Point Methods

- Goal: Solve a linear program

$$\min_{\mathbf{x}\in\mathbb{R}^n} \mathbf{c}^T\mathbf{x} \quad \text{s.t.} \quad \mathbf{M}\mathbf{x} = \mathbf{b}, \;\; \mathbf{x} \geq 0.$$

- Interior-point algorithms stay strictly inside the feasible region, following a central path that leads to the optimum.

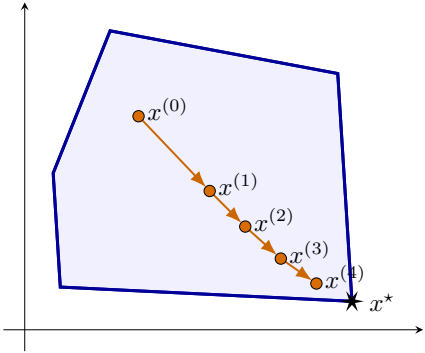


Interior-Point Methods

- Goal: Solve a linear program

$$\min_{\mathbf{x}\in\mathbb{R}^n} \mathbf{c}^T\mathbf{x} \quad \text{s.t.} \quad \mathbf{M}\mathbf{x} = \mathbf{b}, \;\; \mathbf{x} \geq 0.$$

- Interior-point algorithms stay strictly inside the feasible region, following a central path that leads to the optimum.
- Output: an ε-approximate solution



[Lee, Sidford '14]'s IPM

[Lee, Sidford '14]'s IPM

- ε-approximate solution
- $\tilde{O}(\sqrt{n}\log(1/\varepsilon))$ iterations
- Each iteration needs:
  - $\tilde{O}(1)$ matrix-vector multiplications.
  - $\tilde{O}(1)$ linear system solves.
  - Leverage score computation with Johnson-Lindenstrauss.
  - Projection on a mixed norm ball:

$$\arg\max_{||\mathbf{x}||_2+||\ell^{-1}\mathbf{x}||_\infty \leq 1} \mathbf{a}^T\mathbf{x} \qquad \text{for some } \mathbf{a}, \ell \in \mathbb{R}^m.$$

[Lee, Sidford '14]'s IPM

- ε-approximate solution
- $\tilde{O}(\sqrt{n}\log(1/\varepsilon))$ iterations
- Each iteration needs:
  - $\tilde{O}(1)$ matrix-vector multiplications.
  - $\tilde{O}(1)$ linear system solves.
  - Leverage score computation with Johnson-Lindenstrauss.
  - Projection on a mixed norm ball:

$$\arg\max_{||\mathbf{x}||_2+||\ell^{-1}\mathbf{x}||_\infty \leq 1} \mathbf{a}^T\mathbf{x} \qquad \text{for some } \mathbf{a}, \ell \in \mathbb{R}^m.$$

- For flow: can set up LP s.t. OPT is integral [Daitch, Spielman '08]

[Lee, Sidford '14]'s IPM

- ε-approximate solution
- $\tilde{O}(\sqrt{n}\log(1/\varepsilon))$ iterations
- Each iteration needs:
  - $\tilde{O}(1)$ matrix-vector multiplications.
  - $\tilde{O}(1)$ linear system solves.
  - Leverage score computation with Johnson-Lindenstrauss.
  - Projection on a mixed norm ball:

$$\arg\max_{||\mathbf{x}||_2+||\ell^{-1}\mathbf{x}||_\infty \leq 1} \mathbf{a}^T\mathbf{x} \qquad \text{for some } \mathbf{a}, \ell \in \mathbb{R}^m.$$

- For flow: can set up LP s.t. OPT is integral [Daitch, Spielman '08]
  - Solve to precision $1/m^2$, then f_e on each edge must be close to integral

[Lee, Sidford '14]'s IPM

- ε-approximate solution
- $\tilde{O}(\sqrt{n}\log(1/\varepsilon))$ iterations
- Each iteration needs:
  - $\tilde{O}(1)$ matrix-vector multiplications.
  - $\tilde{O}(1)$ linear system solves.
  - Leverage score computation with Johnson-Lindenstrauss.
  - Projection on a mixed norm ball:

$$\arg\max_{||\mathbf{x}||_2+||\ell^{-1}\mathbf{x}||_\infty \leq 1} \mathbf{a}^T\mathbf{x} \qquad \text{for some } \mathbf{a}, \ell \in \mathbb{R}^m.$$

- For flow: can set up LP s.t. OPT is integral [Daitch, Spielman '08]
  - Solve to precision $1/m^2$, then f_e on each edge must be close to integral
  - Round f_e to closest integer

[Lee, Sidford '14]'s IPM

- ε-approximate solution
- $\tilde{O}(\sqrt{n}\log(1/\varepsilon))$ iterations
- Each iteration needs:
  - $\tilde{O}(1)$ matrix-vector multiplications.
  - $\tilde{O}(1)$ linear system solves.
  - Leverage score computation with Johnson-Lindenstrauss.
  - Projection on a mixed norm ball:

$$\arg\max_{||\mathbf{x}||_2+||\ell^{-1}\mathbf{x}||_\infty \leq 1} \mathbf{a}^T\mathbf{x} \qquad \text{for some } \mathbf{a}, \ell \in \mathbb{R}^m.$$

- For flow: can set up LP s.t. OPT is integral [Daitch, Spielman '08]
  - Solve to precision $1/m^2$, then f_e on each edge must be close to integral
  - Round f_e to closest integer
- Sequential $\tilde{O}(\sqrt{n} \cdot m)$ time

[Lee, Sidford '14]'s IPM

- ε-approximate solution
- $\tilde{O}(\sqrt{n}\log(1/\varepsilon))$ iterations
- Each iteration needs:
  - $\tilde{O}(1)$ matrix-vector multiplications.
  - $\tilde{O}(1)$ linear system solves.
  - Leverage score computation with Johnson-Lindenstrauss.
  - Projection on a mixed norm ball:

$$\arg\max_{||\mathbf{x}||_2+||\ell^{-1}\mathbf{x}||_\infty \le 1} \mathbf{a}^T\mathbf{x} \qquad \text{for some } \mathbf{a}, \ell \in \mathbb{R}^m.$$

- For flow: can set up LP s.t. OPT is integral [Daitch, Spielman '08]
  - Solve to precision $1/m^2$, then f_e on each edge must be close to integral
  - Round f_e to closest integer
- Sequential $\tilde{O}(\sqrt{n} \cdot m)$ time
- CONGEST: $\tilde{O}(\sqrt{n} \cdot T_{Laplacian}) = n^{o(1)}\sqrt{n}(\sqrt{n} + D)$ rounds

Distributed Min Cost Flow Results

Model	Time	Reference
CONGEST	$\tilde{\Omega}(\sqrt{n}+D)$	[DSHKKNPPW '11][1]
	$n^{o(1)}(\sqrt{n}+D)$	[GKKLPS '15][1]
	$n^{o(1)}\sqrt{n}(\sqrt{n}+D)$	[**dV** '23]

[1] Undirected, approximate max flow
[2] Max flow, M is maximal weight

Distributed Min Cost Flow Results

Model	Time	Reference
CONGEST	$\tilde{\Omega}(\sqrt{n}+D)$	[DSHKKNPPW '11][1]
	$n^{o(1)}(\sqrt{n}+D)$	[GKKLPS '15][1]
	$n^{o(1)}\sqrt{n}(\sqrt{n}+D)$	[**dV** '23]
BCC	$\tilde{O}(\sqrt{n})$	[F**dV** '22]

[1] Undirected, approximate max flow
[2] Max flow, M is maximal weight

Distributed Min Cost Flow Results

Model	Time	Reference
CONGEST	$\tilde{\Omega}(\sqrt{n}+D)$	[DSHKKNPPW '11][1]
	$n^{o(1)}(\sqrt{n}+D)$	[GKKLPS '15][1]
	$n^{o(1)}\sqrt{n}(\sqrt{n}+D)$	[**dV** '23]
BCC	$\tilde{O}(\sqrt{n})$	[F**dV** '22]
Det CC	$m^{3/7+o(1)}M^{1/7}$	[F**dV** '23][2]

[1] Undirected, approximate max flow

[2] Max flow, M is maximal weight

Distributed Min Cost Flow Results

Model	Time	Reference
CONGEST	$\tilde{\Omega}(\sqrt{n}+D)$	[DSHKKNPPW '11][1]
	$n^{o(1)}(\sqrt{n}+D)$	[GKKLPS '15][1]
	$n^{o(1)}\sqrt{n}(\sqrt{n}+D)$	[**dV** '23]
BCC	$\tilde{O}(\sqrt{n})$	[F**dV** '22]
Det CC	$m^{3/7+o(1)}M^{1/7}$	[F**dV** '23][2]
PRAM	$\tilde{O}(\sqrt{n})$ depth and $\tilde{O}(m+n^{1.5})$ work	[vdBGJ**dV**'25]

[1] Undirected, approximate max flow

[2] Max flow, M is maximal weight

Distributed Min Cost Flow Results

Model	Time	Reference
CONGEST	$\tilde{\Omega}(\sqrt{n}+D)$	[DSHKKNPPW '11][1]
	$n^{o(1)}(\sqrt{n}+D)$	[GKKLPS '15][1]
	$n^{o(1)}\sqrt{n}(\sqrt{n}+D)$	[**dV** '23]
BCC	$\tilde{O}(\sqrt{n})$	[F**dV** '22]
Det CC	$m^{3/7+o(1)}M^{1/7}$	[F**dV** '23][2]
PRAM	$\tilde{O}(\sqrt{n})$ depth and $\tilde{O}(m+n^{1.5})$ work	[vdBGJ**dV**'25]
Sequential	$m^{1+o(1)}$	[CKLPPGS '22]

[1] Undirected, approximate max flow

[2] Max flow, M is maximal weight

Q: Does faster sequential Max Flow transfer to distributed models?

[Lee, Sidford '14]:
#iterations: $\tilde{O}(\sqrt{n})$
Time per iteration: $\tilde{O}(m)$

[Chen, KLPPGS '22]:
#iterations: $m^{1+o(1)}$
Time per iteration: $m^{o(1)}$

Q: Does faster sequential Max Flow transfer to distributed models?

[Lee, Sidford '14]:
#iterations: $\tilde{O}(\sqrt{n})$
Time per iteration: $\tilde{O}(m)$

Iteration count carries over to round complexity

[Chen, KLPPGS '22]:
#iterations: $m^{1+o(1)}$
Time per iteration: $m^{o(1)}$

Running time improvement does not improve round complexity

Q: Does faster sequential Max Flow transfer to distributed models?

[Lee, Sidford '14]:
#iterations: $\tilde{O}(\sqrt{n})$
Time per iteration: $\tilde{O}(m)$

Iteration count carries over to round complexity

[Chen, KLPPGS '22]:
#iterations: $m^{1+o(1)}$
Time per iteration: $m^{o(1)}$

Running time improvement does not improve round complexity

Question

Is $\tilde{\Theta}(\sqrt{n})$ the right iteration count for the min-cost flow LP?

Combinatorial CONGEST Algorithms

- Negative Weighted Shortest Paths:
  - Reduction to positive weight SSSP with $n^{o(1)}$ overhead [Ashvinkumar, Bernstein, Cao, Grunau, Haeupler, Jiang, Nanongkai, and Su '24]
  - Positive SSSP: $n^{o(1)}(n^{2/5}D^{2/5} + \sqrt{n} + D)$ rounds [Cao and Fineman '23, Rozhoň (r) Haeupler (r) Martinsson (r) Grunau (r) Zuzic '23]

Combinatorial CONGEST Algorithms

- Negative Weighted Shortest Paths:
  - Reduction to positive weight SSSP with $n^{o(1)}$ overhead [Ashvinkumar, Bernstein, Cao, Grunau, Haeupler, Jiang, Nanongkai, and Su '24]
  - Positive SSSP: $n^{o(1)}(n^{2/5}D^{2/5} + \sqrt{n} + D)$ rounds [Cao and Fineman '23, Rozhoň (r) Haeupler (r) Martinsson (r) Grunau (r) Zuzic '23]
- Max Flow in Planar Graphs: $\tilde{O}(D^2)$ rounds [Abd-Elhaleem, Dory, Parter, and Weimann '25]

Laplacian Paradigm

- Laplacian systems
- Spectral sparsifiers
- Electrical flow
- Effective resistance
- Sampling spanning trees
- Expander decompositions
- Continuous optimization
- Interior-point methods
- Gradient descent
- Preconditioning
- ...

Conclusion

The Laplacian paradigm

- has many existing tools (not optimal!)
- has many applications
- allows us to use algebraic (sequential) techniques

Conclusion

The Laplacian paradigm

- has many existing tools (not optimal!)
- has many applications
- allows us to use algebraic (sequential) techniques

Laplacian paradigm gives state of the art for flow problems in

- Sequential
- Parallel
- CONGEST
- (Broadcast/Determinisitic) Congested Clique

Conclusion

The Laplacian paradigm

- has many existing tools (not optimal!)
- has many applications
- allows us to use algebraic (sequential) techniques

Laplacian paradigm gives state of the art for flow problems in

- Sequential
- Parallel
- CONGEST
- (Broadcast/Determinisitic) Congested Clique

Thank you!