Linear Sketching and Applications to Distributed Computation

Cameron Musco

November 7, 2014

Overview

Linear Sketches

- Johnson-Lindenstrauss Lemma
- Heavy-Hitters

Applications

- k-Means Clustering
- Spectral sparsification

Overview

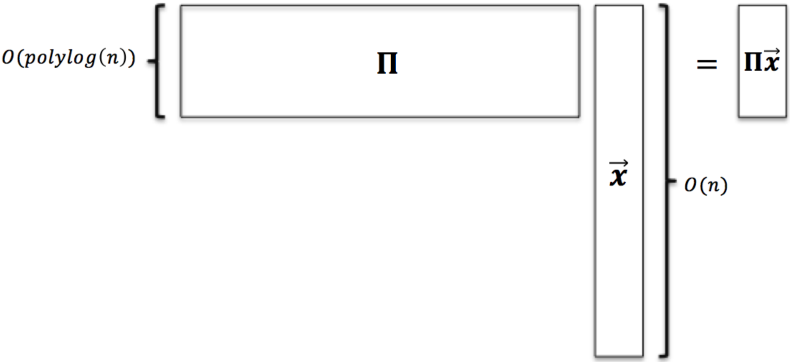
Linear Sketches

- Johnson-Lindenstrauss Lemma
- Heavy-Hitters

Applications

- k-Means Clustering
- Spectral sparsification

Linear Sketching



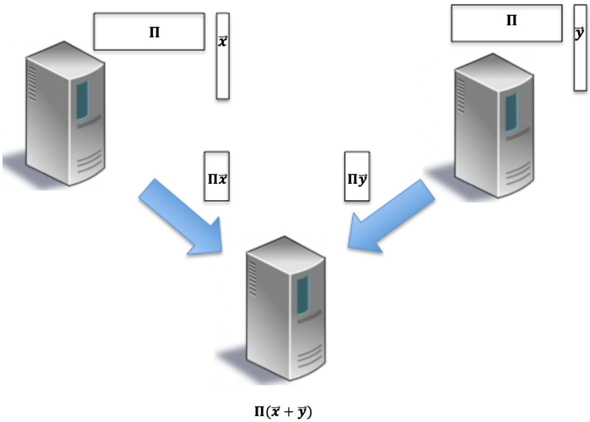
- Randomly choose $\Pi \sim \mathcal{D}$

Linear Sketching

- Oblivious: $\mathbf{\Pi}$ chosen independently of $\mathbf{x}$.
- Composable: $\mathbf{\Pi}(\mathbf{x} + \mathbf{y}) = \mathbf{\Pi}\mathbf{x} + \mathbf{\Pi}\mathbf{y}$.

Linear Sketching

- Oblivious: $\mathbf{\Pi}$ chosen independently of $\mathbf{x}$.
- Composable: $\mathbf{\Pi}(\mathbf{x} + \mathbf{y}) = \mathbf{\Pi}\mathbf{x} + \mathbf{\Pi}\mathbf{y}$.



Linear Sketching

- Streaming algorithms with *polylog*(n) space.
- Frequency moments, heavy hitters, entropy estimation

$$\begin{bmatrix} 1 \\ -2 \\ 0 \\ \vdots \\ 5 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 5 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 5 \\ 6 \end{bmatrix} \rightarrow \ldots \rightarrow \begin{bmatrix} 14 \\ -2 \\ 3 \\ \vdots \\ 0 \\ 4 \end{bmatrix}$$

$$\mathbf{\Pi x_0} \rightarrow \mathbf{\Pi}(\mathbf{x_0} + \mathbf{x_1}) \rightarrow \ldots \rightarrow \mathbf{\Pi}(\mathbf{x_0} + \mathbf{x_1} + \ldots + \mathbf{x_n})$$

Linear Sketching

- Streaming algorithms with *polylog*(n) space.
- Frequency moments, heavy hitters, entropy estimation

$$\begin{bmatrix} 1 \\ -2 \\ 0 \\ \vdots \\ 5 \\ 0 \end{bmatrix} \to \begin{bmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 5 \\ 0 \end{bmatrix} \to \begin{bmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 5 \\ 6 \end{bmatrix} \to \dots \to \begin{bmatrix} 14 \\ -2 \\ 3 \\ \vdots \\ 0 \\ 4 \end{bmatrix}$$

$$\mathbf{\Pi x}_0 \to \mathbf{\Pi}(\mathbf{x}_0 + \mathbf{x}_1) \to \dots \to \mathbf{\Pi}(\mathbf{x}_0 + \mathbf{x}_1 + \dots + \mathbf{x_n})$$

Linear Sketching

- Streaming algorithms with *polylog*(n) space.
- Frequency moments, heavy hitters, entropy estimation

$$\begin{bmatrix} 1 \\ -2 \\ 0 \\ \vdots \\ 5 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 5 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 5 \\ 6 \end{bmatrix} \rightarrow \ldots \rightarrow \begin{bmatrix} 14 \\ -2 \\ 3 \\ \vdots \\ 0 \\ 4 \end{bmatrix}$$

$$\mathbf{\Pi x_0} \rightarrow \mathbf{\Pi(x_0 + x_1)} \rightarrow \ldots \rightarrow \mathbf{\Pi(x_0 + x_1 + \ldots + x_n)}$$

Linear Sketching

- Streaming algorithms with *polylog*(n) space.
- Frequency moments, heavy hitters, entropy estimation

$$\begin{bmatrix} 1 \\ -2 \\ 0 \\ \vdots \\ 5 \\ 0 \end{bmatrix} \to \begin{bmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 5 \\ 0 \end{bmatrix} \to \begin{bmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 5 \\ 6 \end{bmatrix} \to \dots \to \begin{bmatrix} 14 \\ -2 \\ 3 \\ \vdots \\ 0 \\ 4 \end{bmatrix}$$

$$\mathbf{\Pi x_0} \to \mathbf{\Pi(x_0 + x_1)} \to \dots \to \mathbf{\Pi(x_0 + x_1 + \dots + x_n)}$$

Linear Sketching

- Streaming algorithms with *polylog*(n) space.
- Frequency moments, heavy hitters, entropy estimation

$$\begin{bmatrix} 1 \\ -2 \\ 0 \\ \vdots \\ 5 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 5 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 5 \\ 6 \end{bmatrix} \rightarrow \ldots \rightarrow \begin{bmatrix} 14 \\ -2 \\ 3 \\ \vdots \\ 0 \\ 4 \end{bmatrix}$$

$$\mathbf{\Pi x_0} \rightarrow \mathbf{\Pi}(\mathbf{x_0} + \mathbf{x_1}) \rightarrow \ldots \rightarrow \mathbf{\Pi}(\mathbf{x_0} + \mathbf{x_1} + \ldots + \mathbf{x_n})$$

Linear Sketching

- Streaming algorithms with *polylog*(n) space.
- Frequency moments, heavy hitters, entropy estimation

$$\begin{bmatrix} 1 \\ -2 \\ 0 \\ \vdots \\ 5 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 5 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 5 \\ 6 \end{bmatrix} \rightarrow \dots \rightarrow \begin{bmatrix} 14 \\ -2 \\ 3 \\ \vdots \\ 0 \\ 4 \end{bmatrix}$$

$$\mathbf{\Pi x}_0 \rightarrow \mathbf{\Pi}(\mathbf{x}_0 + \mathbf{x}_1) \rightarrow \dots \rightarrow \mathbf{\Pi}(\mathbf{x}_0 + \mathbf{x}_1 + \dots + \mathbf{x_n})$$

Linear Sketching

Two Main Tools

- Johnson Lindenstrauss sketches (randomized dimensionality reduction, subspace embedding, etc..)
- Heavy-Hitters sketches (sparse recovery, compressive sensing, ℓ_p sampling, graph sketching, etc....)

Johnson-Lindenstrauss Lemma

- Low Dimensional Embedding. $n \to m = O(\log(1/\delta)/\epsilon^2)$
- $\|\mathbf{\Pi x}\|_2^2 \approx_\epsilon \|\mathbf{x}\|_2^2$.

$$\frac{1}{\sqrt{m}} \begin{bmatrix} \mathcal{N}(0,1) & \dots & \mathcal{N}(0,1) \\ \vdots & & \vdots \\ \mathcal{N}(0,1) & \dots & \mathcal{N}(0,1) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \frac{1}{\sqrt{m}} \begin{bmatrix} \mathcal{N}(0, x_1^2 + \dots + x_n^2) \\ \vdots \\ \mathcal{N}(0, x_1^2 + \dots + x_n^2) \end{bmatrix}$$

Johnson-Lindenstrauss Lemma

- Low Dimensional Embedding. $n \to m = O(\log(1/\delta)/\epsilon^2)$
- $\|\mathbf{\Pi x}\|_2^2 \approx_\epsilon \|\mathbf{x}\|_2^2$.

$$\frac{1}{\sqrt{m}} \begin{bmatrix} \mathcal{N}(0,1) & \dots & \mathcal{N}(0,1) \\ \vdots & & \vdots \\ \mathcal{N}(0,1) & \dots & \mathcal{N}(0,1) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \frac{1}{\sqrt{m}} \begin{bmatrix} \mathcal{N}(0, x_1^2 + \dots + x_n^2) \\ \vdots \\ \mathcal{N}(0, x_1^2 + \dots + x_n^2) \end{bmatrix}$$

Johnson-Lindenstrauss Lemma

- Low Dimensional Embedding. $n \to m = O(\log(1/\delta)/\epsilon^2)$
- $\|\mathbf{\Pi x}\|_2^2 \approx_\epsilon \|\mathbf{x}\|_2^2$.

$$\frac{1}{\sqrt{m}} \begin{bmatrix} \mathcal{N}(0,1) & \dots & \mathcal{N}(0,1) \\ \vdots & & \vdots \\ \mathcal{N}(0,1) & \dots & \mathcal{N}(0,1) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \frac{1}{\sqrt{m}} \begin{bmatrix} \mathcal{N}(0, \|\mathbf{x}\|_2^2) \\ \vdots \\ \mathcal{N}(0, \|\mathbf{x}\|_2^2) \end{bmatrix}$$

$$\|\mathbf{\Pi x}\|_2^2 = \frac{1}{m} \sum_{i=1}^{m} \mathcal{N}(0, \|\mathbf{x}\|_2^2)^2 \approx_\epsilon \|\mathbf{x}\|_2^2$$

Johnson-Lindenstrauss Lemma

- Low Dimensional Embedding. $n \to m = O(\log(1/\delta)/\epsilon^2)$
- $\|\mathbf{\Pi x}\|_2^2 \approx_\epsilon \|\mathbf{x}\|_2^2$.

$$\frac{1}{\sqrt{m}} \begin{bmatrix} \mathcal{N}(0,1) & \dots & \mathcal{N}(0,1) \\ \vdots & & \vdots \\ \mathcal{N}(0,1) & \dots & \mathcal{N}(0,1) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \frac{1}{\sqrt{m}} \begin{bmatrix} \mathcal{N}(0, \|\mathbf{x}\|_2^2) \\ \vdots \\ \mathcal{N}(0, \|\mathbf{x}\|_2^2) \end{bmatrix}$$

$$\|\mathbf{\Pi x}\|_2^2 = \frac{1}{m} \sum_{i=1}^{m} \mathcal{N}(0, \|\mathbf{x}\|_2^2)^2 \approx_\epsilon \|\mathbf{x}\|_2^2$$

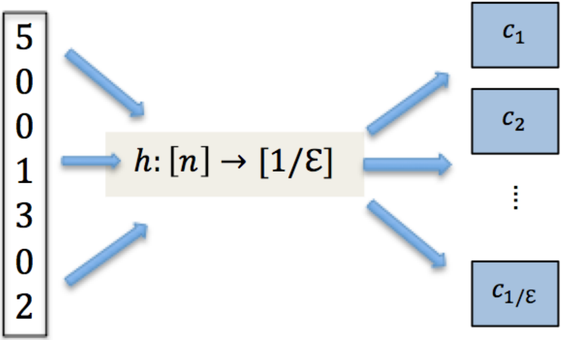
Johnson-Lindenstrauss Lemma

That's it! - basic statistics

- Sparse constructions.
- ± 1 replace Gaussians
- Small sketch representation - i.e. small random seed (otherwise storing $\mathbf{\Pi}$ takes more space than storing $\mathbf{x}$)

Heavy-Hitters

- Count sketch, sparse recovery, ℓ_p sampling, point query, graph sketching, sparse fourier transform
- Simple idea: Hashing



Heavy-Hitters

- Random signs to deal with negative entries
- Repeat many times 'decode' heavy buckets

$$\begin{bmatrix} 1 & 0 & 0 & \dots & -1 & 0 \\ \vdots & & & & & \vdots \\ 0 & -1 & 0 & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \color{blue}{x_2} \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \color{blue}{c_1} \\ \vdots \\ c_{1/\epsilon} \end{bmatrix}$$

$$\vdots$$

$$\begin{bmatrix} 1 & 0 & 0 & \dots & -1 & 0 \\ \vdots & & & & & \vdots \\ 0 & -1 & 0 & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \color{blue}{x_2} \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \vdots \\ \color{blue}{c_5} \\ \vdots \\ c_{1/\epsilon} \end{bmatrix}$$

Heavy-Hitters

- Random signs to deal with negative entries
- Repeat many times 'decode' heavy buckets

$$\begin{bmatrix} c_1 \\ \vdots \\ \vdots \\ c_{1/\epsilon} \end{bmatrix}, \begin{bmatrix} \vdots \\ c_5 \\ \vdots \\ c_{1/\epsilon} \end{bmatrix}, \begin{bmatrix} \vdots \\ c_{20} \\ \vdots \\ c_{1/\epsilon} \end{bmatrix}, \begin{bmatrix} \vdots \\ \vdots \\ \vdots \\ c_{1/\epsilon} \end{bmatrix}$$

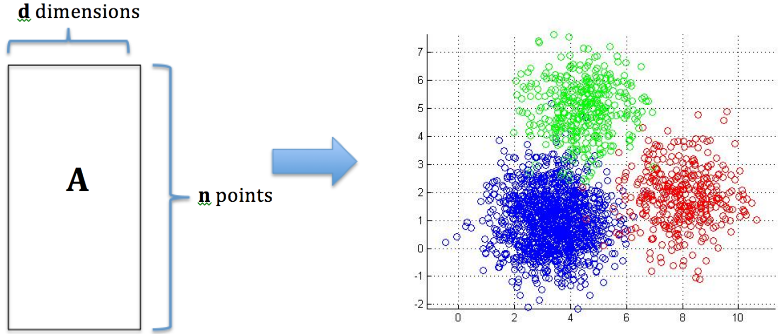
- $h_1(2) = 1, h_2(2) = 5, h_3(2) = 20, h_4(2) = 1/\epsilon \rightarrow \frac{x_2^2}{\|\mathbf{x}\|_2^2} \geq \frac{1}{\epsilon}$

Heavy-Hitters

- Just random encoding
- *polylog*(n) to recover entires with $\frac{1}{\log(n)}$ of the total norm
- Basically best we could hope for.
- Random scalings gives ℓ_p sampling.

Application 1: k-means Clustering

- Assign points to k clusters
- k is fixed
- Minimize distance to centroids: $\sum_{i=1}^{n} \|\mathbf{a}_i - \boldsymbol{\mu}_{C(i)}\|_2^2$



Application 1: k-means Clustering

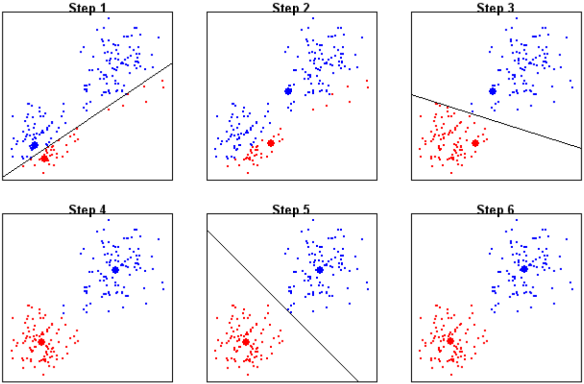
Lloyd's algorithm aka the 'k-means algorithm'

- Initalize random clusters
- Compute centroids
- Assign each point to closest centroid
- Repeat

Application 1: k-means Clustering

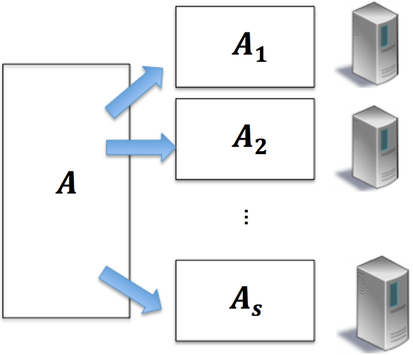
Lloyd's algorithm aka the 'k-means algorithm'

- Initalize random clusters
- Compute centroids
- Assign each point to closest centroid
- Repeat



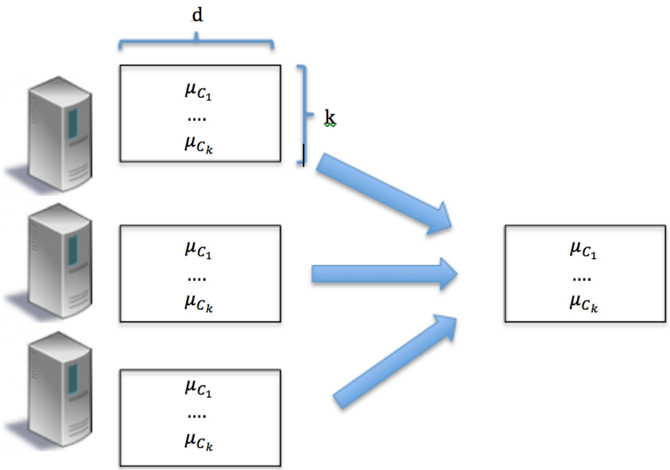
Application 1: k-means Clustering

What if data is distributed?



Application 1: k-means Clustering

- At each iteration each server sends out new local centroids
- Adding them together, gives the new global centroids.
- $O(sdk)$ communication per iteration



Application 1: k-means Clustering

Can we do better?

- Balcan et al. 2013 - $\tilde{O}((kd + sk)d)$
- Locally computable $\tilde{O}(kd + sk)$ sized coreset.
- All data is aggregated and k-means performed on single server.

Can we do even better?

Can we do even better?

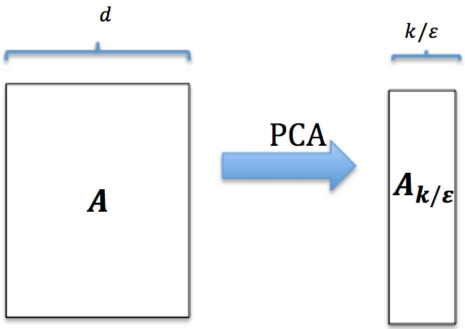
- $O((sd + sk)d) \rightarrow O((sd' + sk)d')$ for $d' << d$.

Can we do even better?

- $O((sd + sk)d) \to O((sd' + sk)d')$ for $d' << d$.
- Liang et al. 2013, Balcan et al. 2014 - $\tilde{O}((sk + sk)k + sdk)$

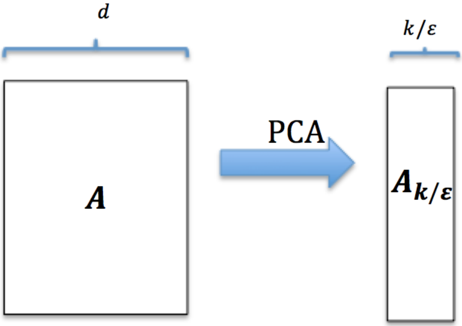
Can we do even better?

- $O((sd + sk)d) \to O((sd' + sk)d')$ for $d' << d$.
- Liang et al. 2013, Balcan et al. 2014 - $\tilde{O}((sk + sk)k + sdk)$



Can we do even better?

- $O((sd + sk)d) \to O((sd' + sk)d')$ for $d' << d$.
- Liang et al. 2013, Balcan et al. 2014 - $\tilde{O}((sk + sk)k + sdk)$



- CEMMP '14: improved $O(k/\epsilon^2)$ to $\lceil k/\epsilon \rceil$.

Application 1: k-means Clustering

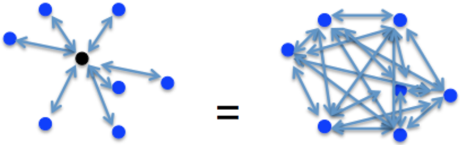
Can we do better?

- $O(sdk)$ inherent in communicating $O(k)$ singular vectors of dimension d to s servers.
- Apply Johnson-Lindenstrauss!

Application 1: k-means Clustering

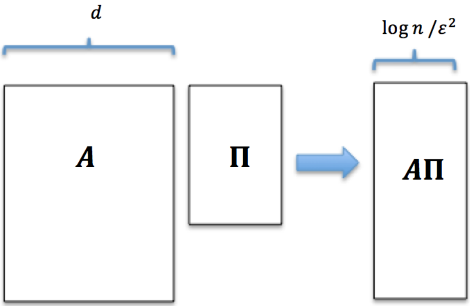
Can we do better?

- $O(sdk)$ inherent in communicating $O(k)$ singular vectors of dimension d to s servers.
- Apply Johnson-Lindenstrauss!
- Goal: Minimize distance to centroids: $\sum_{i=1}^{n} \|\mathbf{a}_i - \boldsymbol{\mu}_{C(i)}\|_2^2$



- Equivalently all pairs distances: $\sum_{i=1}^{k} \frac{1}{|C_i|} \sum_{(j,k)\in C_i} \|\mathbf{a}_i - \mathbf{a}_j\|_2^2$

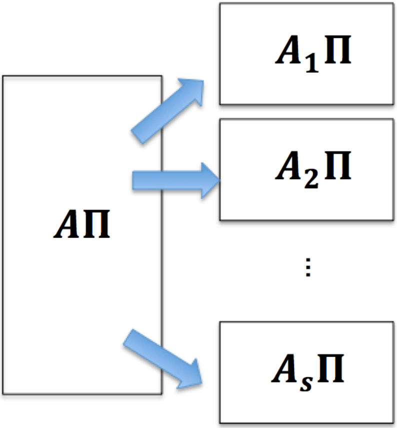
Application 1: k-means Clustering



- $\|\mathbf{a}_i - \mathbf{a}_j\|_2^2 \approx_\epsilon \|(\mathbf{a}_i - \mathbf{a}_j)\mathbf{\Pi}\|_2^2 = \|\mathbf{a}_i\mathbf{\Pi} - \mathbf{a}_j\mathbf{\Pi}\|_2^2$
- $O(n^2)$ distance vectors so set failure probability $\delta = \frac{1}{100 * n^2}$.
- $\mathbf{\Pi}$ needs $O(\log 1/\delta/\epsilon^2) = O(\log n/\epsilon^2)$ dimensions

Application 1: k-means Clustering

Immediately distributes - just need to share randomness specifying $\mathbf{\Pi}$.



Application 1: k-means Clustering

Our Paper [Cohen Elder Musco Musco Persu 14]

- Show that $\mathbf{\Pi}$ only needs to have $O(k/\epsilon^2)$ columns
- Almost completely removes dependence on input size!
- $\tilde{O}(k^3 + sk^2 + \log d)$ - $\log d$ gets swallowed in the word size.

Application 1: k-means Clustering

Highest Level Idea for how this works

- Show that the cost of projecting the columns $\mathbf{A\Pi}$ to any k-dimensional subspace approximates the cost of projecting $\mathbf{A}$ to that subspace.
- Note that k-means can actually be viewed as a column projection problem.
- k-means clustering is 'constrained' PCA
- Lots of applications aside from k-means clustering.

Application 1: k-means Clustering

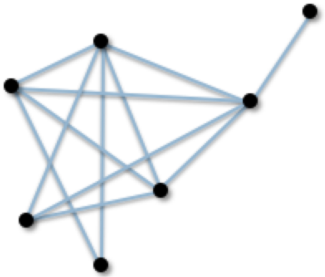
Open Questions

- $(9+\epsilon)$-approximation with only $O(\log k)$ dimensions! What is the right answer?
- We use $\tilde{O}(kd + sk)$ sized coresets blackbox and reduce d. Can we use our linear algebraic understanding to improve coreset constructions? I feels like we should be able to.
- These algorithms should be practical. I think testing them out would be useful - for both k-means and PCA.
- Other problems (spectral clustering, SVM, what do people actually do?)

Application 2: Spectral Sparsification

General Idea

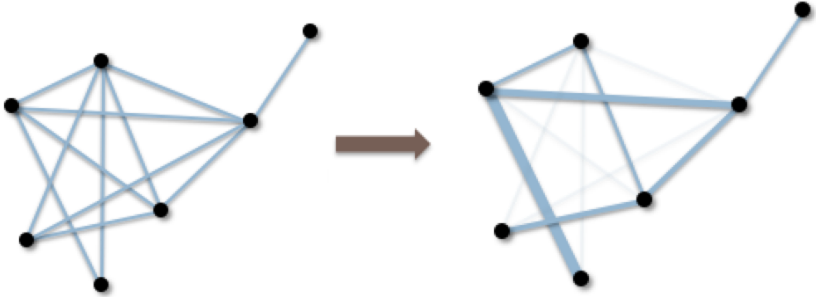
- Approximate a dense graph with a much sparser graph.
- Reduce $O(n^2)$ edges $\rightarrow$ $O(n \log n)$ edges



Application 2: Spectral Sparsification

General Idea

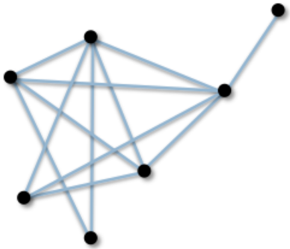
- Approximate a dense graph with a much sparser graph.
- Reduce $O(n^2)$ edges $\rightarrow O(n \log n)$ edges

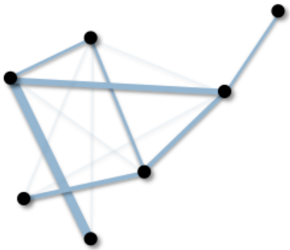


Application 2: Spectral Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Preserve *every* cut value to within $(1 \pm \varepsilon)$ factor



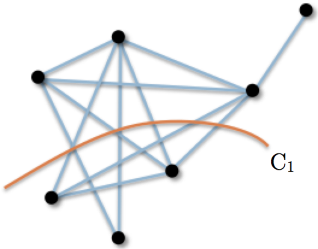


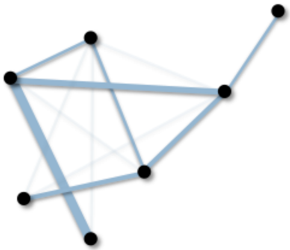
Applications: Minimum cut, sparsest cut, etc.

Application 2: Spectral Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Preserve *every* cut value to within $(1 \pm \varepsilon)$ factor



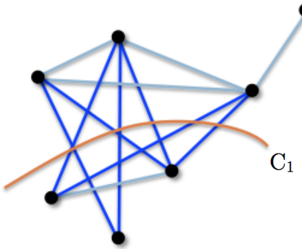


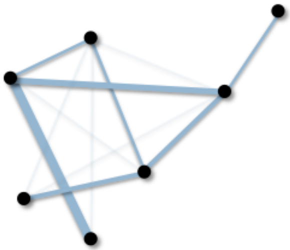
Applications: Minimum cut, sparsest cut, etc.

Application 2: Spectral Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Preserve *every* cut value to within $(1 \pm \varepsilon)$ factor



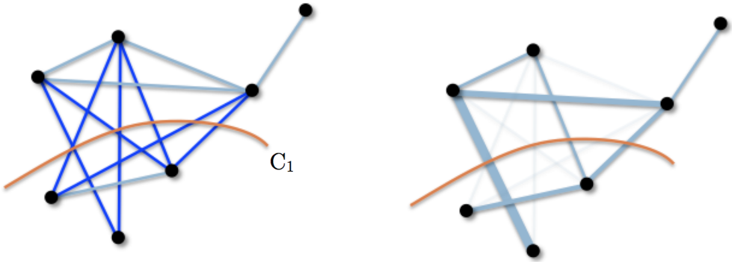


Applications: Minimum cut, sparsest cut, etc.

Application 2: Spectral Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Preserve *every* cut value to within $(1 \pm \varepsilon)$ factor

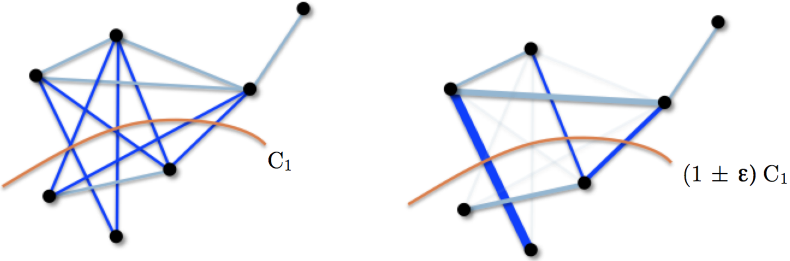


Applications: Minimum cut, sparsest cut, etc.

Application 2: Spectral Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Preserve *every* cut value to within $(1 \pm \varepsilon)$ factor

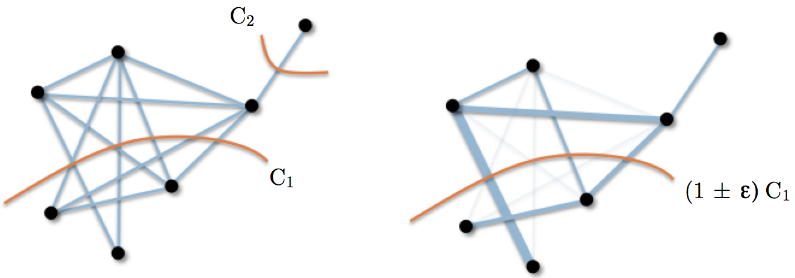


Applications: Minimum cut, sparsest cut, etc.

Application 2: Spectral Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Preserve *every* cut value to within $(1 \pm \varepsilon)$ factor

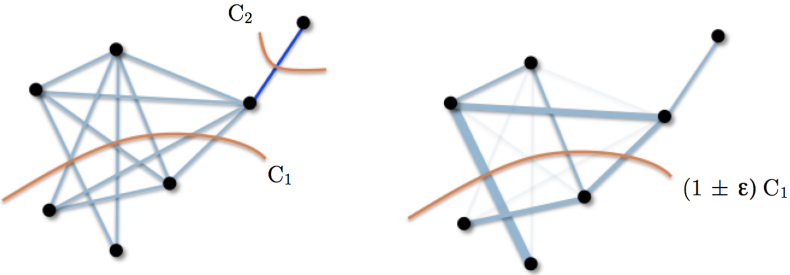


Applications: Minimum cut, sparsest cut, etc.

Application 2: Spectral Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Preserve *every* cut value to within $(1 \pm \varepsilon)$ factor

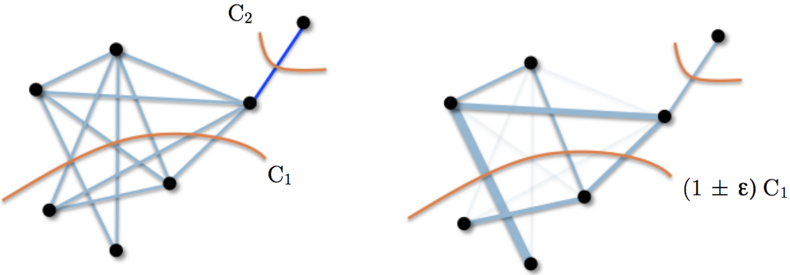


Applications: Minimum cut, sparsest cut, etc.

Application 2: Spectral Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Preserve *every* cut value to within $(1 \pm \varepsilon)$ factor

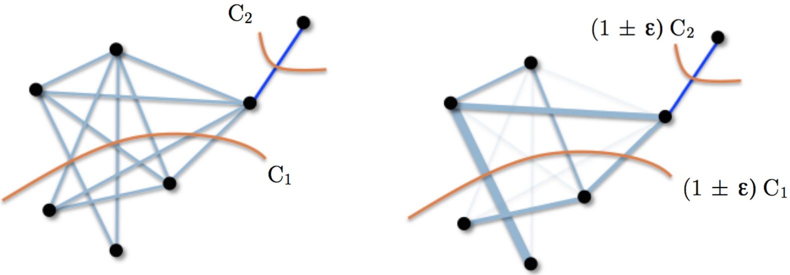


Applications: Minimum cut, sparsest cut, etc.

Application 2: Spectral Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Preserve *every* cut value to within $(1 \pm \varepsilon)$ factor

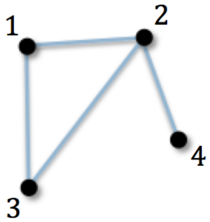


Applications: Minimum cut, sparsest cut, etc.

Graph Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Let $\mathbf{B} \in \mathbb{R}^{\binom{n}{2} \times n}$ be the vertex-edge incidence matrix for a graph G.
- Let $\mathbf{x} \in \{0,1\}^n$ be an "indicator vector" for some cut.

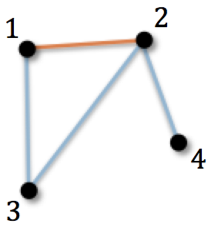


$$\begin{array}{c} \\ e_{12} \\ e_{13} \\ e_{14} \\ e_{23} \\ e_{24} \\ e_{34} \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ \mathbf{B} \end{array}$$

Graph Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Let $\mathbf{B} \in \mathbb{R}^{\binom{n}{2} \times n}$ be the vertex-edge incidence matrix for a graph G.
- Let $\mathbf{x} \in \{0, 1\}^n$ be an "indicator vector" for some cut.

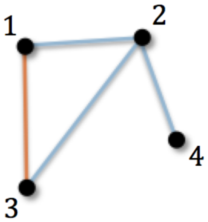


$$\begin{array}{c} \\ e_{12} \\ e_{13} \\ e_{14} \\ e_{23} \\ e_{24} \\ e_{34} \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} \mathbf{1} & \mathbf{-1} & \mathbf{0} & \mathbf{0} \\ 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ \mathbf{B} \end{array}$$

Graph Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Let $\mathbf{B} \in \mathbb{R}^{\binom{n}{2} \times n}$ be the vertex-edge incidence matrix for a graph G.
- Let $\mathbf{x} \in \{0,1\}^n$ be an "indicator vector" for some cut.



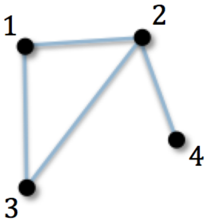
$$\begin{array}{c c} & \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{array}{c} e_{12} \\ e_{13} \\ e_{14} \\ e_{23} \\ e_{24} \\ e_{34} \end{array} & \begin{bmatrix} 1 & -1 & 0 & 0 \\ \mathbf{1} & \mathbf{0} & \mathbf{-1} & \mathbf{0} \\ 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$\mathbf{B}$$

Graph Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Let $\mathbf{B} \in \mathbb{R}^{\binom{n}{2} \times n}$ be the vertex-edge incidence matrix for a graph G.
- Let $\mathbf{x} \in \{0,1\}^n$ be an "indicator vector" for some cut.

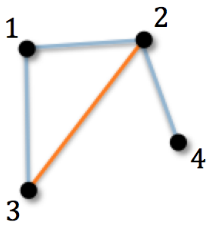


$$
\begin{array}{c}
\\
e_{12}\\ e_{13}\\ e_{14}\\ e_{23}\\ e_{24}\\ e_{34}
\end{array}
\begin{array}{c}
\begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array}\\
\begin{bmatrix}
1 & -1 & 0 & 0\\
1 & 0 & -1 & 0\\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\
0 & 1 & -1 & 0\\
0 & 1 & 0 & -1\\
0 & 0 & 0 & 0
\end{bmatrix}\\
\mathbf{B}
\end{array}
$$

Graph Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Let $\mathbf{B} \in \mathbb{R}^{\binom{n}{2} \times n}$ be the vertex-edge incidence matrix for a graph G.
- Let $\mathbf{x} \in \{0,1\}^n$ be an "indicator vector" for some cut.



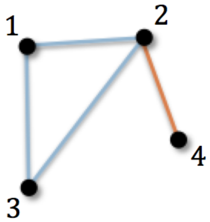
$$
\begin{array}{c} \\ e_{12} \\ e_{13} \\ e_{14} \\ e_{23} \\ e_{24} \\ e_{34} \end{array}
\begin{array}{c}
\begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\
\begin{bmatrix}
1 & -1 & 0 & 0 \\
1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 \\
\mathbf{0} & \mathbf{1} & \mathbf{-1} & \mathbf{0} \\
0 & 1 & 0 & -1 \\
0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

$\mathbf{B}$

Graph Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Let $\mathbf{B} \in \mathbb{R}^{\binom{n}{2} \times n}$ be the vertex-edge incidence matrix for a graph G.
- Let $\mathbf{x} \in \{0,1\}^n$ be an "indicator vector" for some cut.



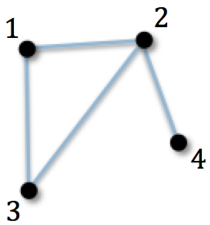
$$\begin{array}{c c} & \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{array}{c} e_{12} \\ e_{13} \\ e_{14} \\ e_{23} \\ e_{24} \\ e_{34} \end{array} & \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{-0} \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$\mathbf{B}$$

Graph Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Let $\mathbf{B} \in \mathbb{R}^{\binom{n}{2} \times n}$ be the vertex-edge incidence matrix for a graph G.
- Let $\mathbf{x} \in \{0,1\}^n$ be an "indicator vector" for some cut.



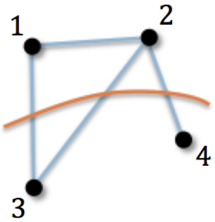
$$
\begin{array}{c|cccc|}
 & v_1 & v_2 & v_3 & v_4 \\
\hline
e_{12} & 1 & -1 & 0 & 0 \\
e_{13} & 1 & 0 & -1 & 0 \\
e_{14} & 0 & 0 & 0 & 0 \\
e_{23} & 0 & 1 & -1 & 0 \\
e_{24} & 0 & 1 & 0 & -1 \\
e_{34} & 0 & 0 & 0 & 0 \\
\hline
\end{array}
$$

$\mathbf{B}$

Graph Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Let $\mathbf{B} \in \mathbb{R}^{\binom{n}{2} \times n}$ be the vertex-edge incidence matrix for a graph G.
- Let $\mathbf{x} \in \{0,1\}^n$ be an "indicator vector" for some cut.

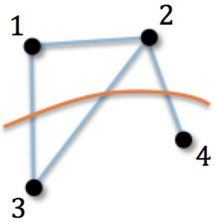


$$
\begin{array}{cc}
 & \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\
\begin{array}{c} e_{12} \\ e_{13} \\ e_{14} \\ e_{23} \\ e_{24} \\ e_{34} \end{array} &
\begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
 & \mathbf{B}
\end{array}
\times
\begin{array}{c}
\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\
\mathbf{x}
\end{array}
$$

Graph Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Let $\mathbf{B} \in \mathbb{R}^{\binom{n}{2} \times n}$ be the vertex-edge incidence matrix for a graph G.
- Let $\mathbf{x} \in \{0,1\}^n$ be an "indicator vector" for some cut.

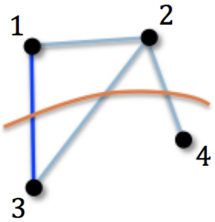


$$
\underbrace{\begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline e_{12} & 1 & -1 & 0 & 0 \\ e_{13} & 1 & 0 & -1 & 0 \\ e_{14} & 0 & 0 & 0 & 0 \\ e_{23} & 0 & 1 & -1 & 0 \\ e_{24} & 0 & 1 & 0 & -1 \\ e_{34} & 0 & 0 & 0 & 0 \end{array}}_{\mathbf{B}} \times \underbrace{\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{x}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}
$$

Graph Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Let $\mathbf{B} \in \mathbb{R}^{\binom{n}{2} \times n}$ be the vertex-edge incidence matrix for a graph G.
- Let $\mathbf{x} \in \{0,1\}^n$ be an "indicator vector" for some cut.

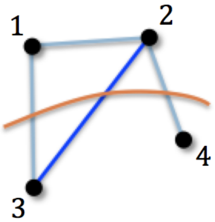


$$\begin{array}{c} \\ e_{12} \\ e_{13} \\ e_{14} \\ e_{23} \\ e_{24} \\ e_{34} \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ \mathbf{B} \end{array} \times \underset{\mathbf{x}}{\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Graph Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Let $\mathbf{B} \in \mathbb{R}^{\binom{n}{2} \times n}$ be the vertex-edge incidence matrix for a graph G.
- Let $\mathbf{x} \in \{0,1\}^n$ be an "indicator vector" for some cut.

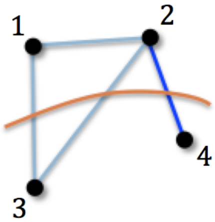


$$\underset{\mathbf{B}}{\begin{array}{cc} & \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{array}{c} e_{12} \\ e_{13} \\ e_{14} \\ e_{23} \\ e_{24} \\ e_{34} \end{array} & \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}} \times \underset{\mathbf{x}}{\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ {\color{blue}1} \\ 1 \\ 0 \end{bmatrix}$$

Graph Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Let $\mathbf{B} \in \mathbb{R}^{\binom{n}{2} \times n}$ be the vertex-edge incidence matrix for a graph G.
- Let $\mathbf{x} \in \{0,1\}^n$ be an "indicator vector" for some cut.

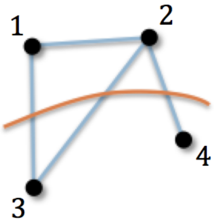


$$\begin{array}{c} \\ e_{12} \\ e_{13} \\ e_{14} \\ e_{23} \\ e_{24} \\ e_{34} \end{array} \underbrace{\begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \\ \hline \end{array}}_{} \hspace{-4em} \underbrace{\begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{B}} \times \underbrace{\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{x}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ \color{blue}{1} \\ 0 \end{bmatrix}$$

Graph Sparsification

Cut Sparsification (Benczúr, Karger '96)

- Let $\mathbf{B} \in \mathbb{R}^{\binom{n}{2} \times n}$ be the vertex-edge incidence matrix for a graph G.
- Let $\mathbf{x} \in \{0,1\}^n$ be an "indicator vector" for some cut.



$\|\mathbf{Bx}\|_2^2 = \text{cut value}$

$$\begin{array}{c c} & \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{array}{c} e_{12} \\ e_{13} \\ e_{14} \\ e_{23} \\ e_{24} \\ e_{34} \end{array} & \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ & \mathbf{B} \end{array} \times \begin{array}{c} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ \mathbf{x} \end{array} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Graph Sparsification

Cut Sparsification (Benczúr, Karger '96) So, $\|\mathbf{B}\mathbf{x}\|_2^2 =$ cut value.

Goal

Find some $\tilde{\mathbf{B}}$ such that, for all $\mathbf{x} \in \{0,1\}^n$,

$$(1-\varepsilon)\|\mathbf{B}\mathbf{x}\|_2^2 \leq \|\tilde{\mathbf{B}}\mathbf{x}\|_2^2 \leq (1+\varepsilon)\|\mathbf{B}\mathbf{x}\|_2^2$$

- $\mathbf{x}^\top \tilde{\mathbf{B}}^\top \tilde{\mathbf{B}}\mathbf{x} \approx \mathbf{x}^\top \mathbf{B}^\top \mathbf{B}\mathbf{x}$.
- $\mathbf{L} = \mathbf{B}^\top \mathbf{B}$ is the *graph Laplacian*.

Graph Sparsification

Cut Sparsification (Benczúr, Karger '96) So, $\|\mathbf{Bx}\|_2^2 =$ cut value.

Goal

Find some $\tilde{\mathbf{B}}$ such that, for all $\mathbf{x} \in \{0,1\}^n$,

$$(1-\varepsilon)\|\mathbf{Bx}\|_2^2 \leq \|\tilde{\mathbf{B}}\mathbf{x}\|_2^2 \leq (1+\varepsilon)\|\mathbf{Bx}\|_2^2$$

- $\mathbf{x}^\top \tilde{\mathbf{B}}^\top \tilde{\mathbf{B}} \mathbf{x} \approx \mathbf{x}^\top \mathbf{B}^\top \mathbf{B} \mathbf{x}$.
- $\mathbf{L} = \mathbf{B}^\top \mathbf{B}$ is the *graph Laplacian*.

Graph Sparsification

Spectral Sparsification (Spielman, Teng '04)

Goal

Find some $\tilde{\mathbf{B}}$ such that, for all $\mathbf{x} \in \cancel{\{0,1\}^n}\ \mathbb{R}^n$,

$$(1-\varepsilon)\|\mathbf{Bx}\|_2^2 \leq \|\tilde{\mathbf{B}}\mathbf{x}\|_2^2 \leq (1+\varepsilon)\|\mathbf{Bx}\|_2^2$$

Applications: Anything cut sparsifiers can do, Laplacian system solves, computing effective resistances, spectral clustering, calculating random walk properties, etc.

Graph Sparsification

Spectral Sparsification (Spielman, Teng '04)

Goal

Find some $\tilde{\mathbf{B}}$ such that, for all $\mathbf{x} \in \cancel{\{0,1\}^n}\ \mathbb{R}^n$,

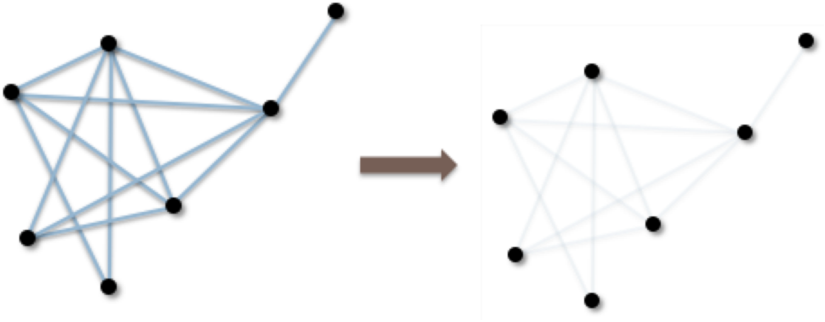
$$(1-\varepsilon)\|\mathbf{B}\mathbf{x}\|_2^2 \leq \|\tilde{\mathbf{B}}\mathbf{x}\|_2^2 \leq (1+\varepsilon)\|\mathbf{B}\mathbf{x}\|_2^2$$

Applications: Anything cut sparsifiers can do, Laplacian system solves, computing effective resistances, spectral clustering, calculating random walk properties, etc.

Graph Sparsification

How are sparsifiers constructed?

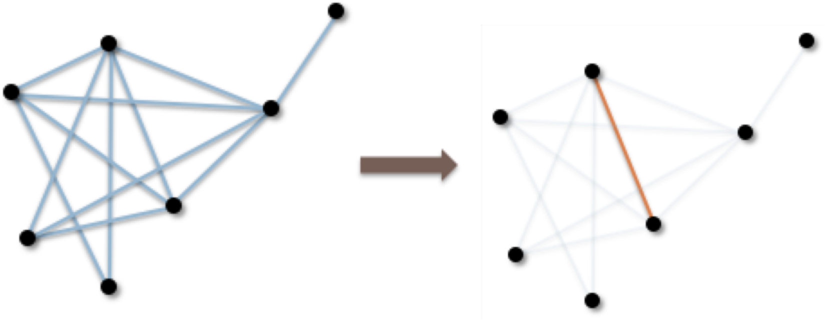
Randomly sample edges (i.e. rows from $\mathbf{B}$):



Graph Sparsification

How are sparsifiers constructed?

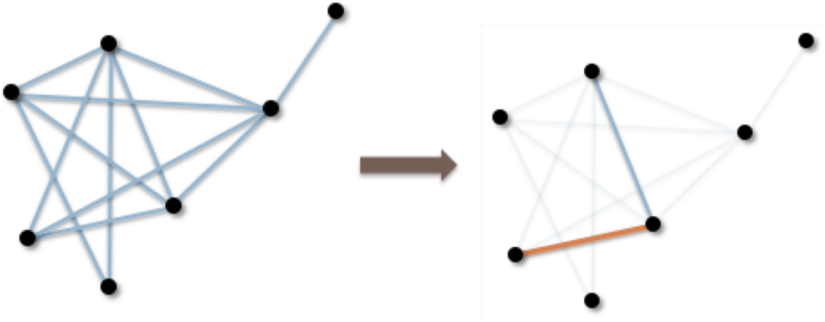
Randomly sample edges (i.e. rows from $\mathbf{B}$):



Graph Sparsification

How are sparsifiers constructed?

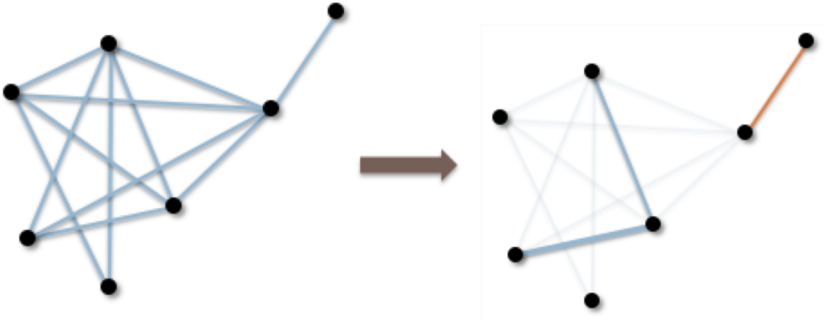
Randomly sample edges (i.e. rows from $\mathbf{B}$):



Graph Sparsification

How are sparsifiers constructed?

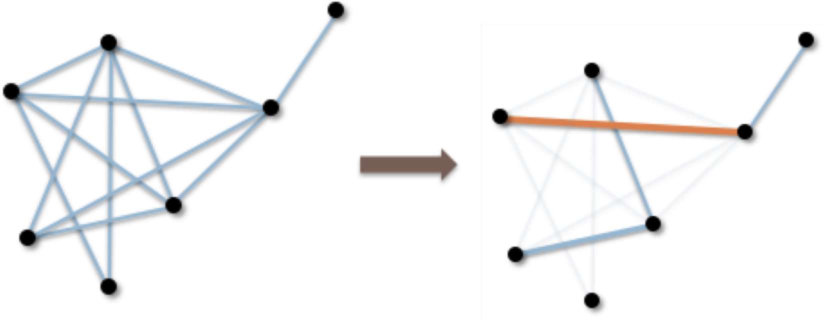
Randomly sample edges (i.e. rows from $\mathbf{B}$):



Graph Sparsification

How are sparsifiers constructed?

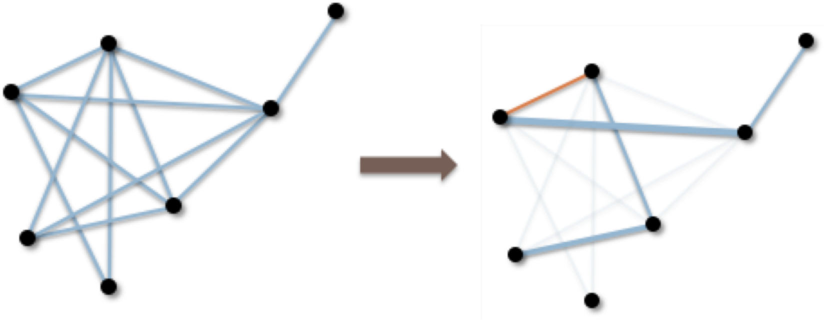
Randomly sample edges (i.e. rows from $\mathbf{B}$):



Graph Sparsification

How are sparsifiers constructed?

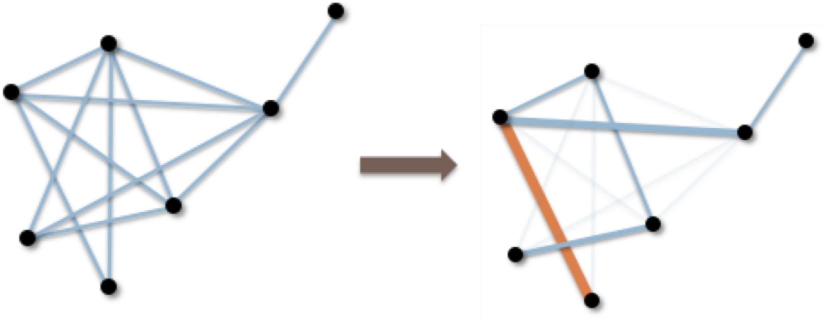
Randomly sample edges (i.e. rows from $\mathbf{B}$):



Graph Sparsification

How are sparsifiers constructed?

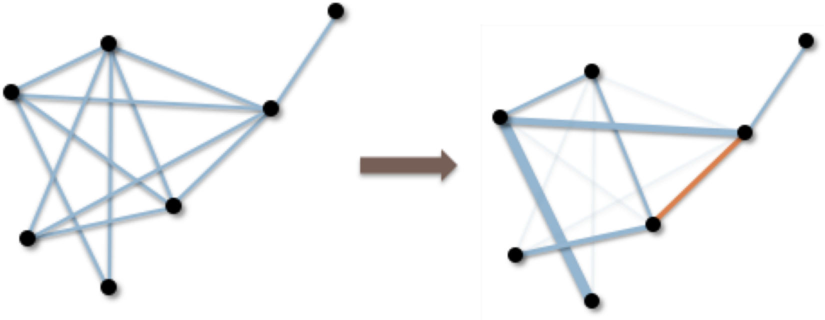
Randomly sample edges (i.e. rows from $\mathbf{B}$):



Graph Sparsification

How are sparsifiers constructed?

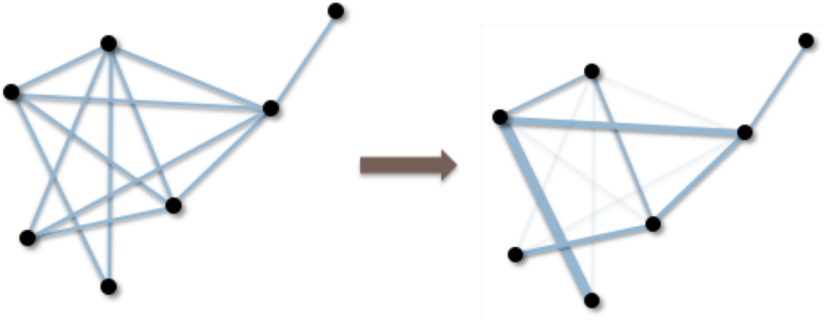
Randomly sample edges (i.e. rows from $\mathbf{B}$):



Graph Sparsification

How are sparsifiers constructed?

Randomly sample edges (i.e. rows from $\mathbf{B}$):



Graph Sparsification

How are sparsifiers constructed?

Sampling probabilities:

- Connectivity for cut sparsifiers [Benczúr, Karger '96], [Fung, Hariharan, Harvey, Panigrahi '11].
- **Effective resistances** (i.e statistical leverage scores) for spectral sparsifiers [Spielman, Srivastava '08].

Actually oversample: by (effective resistance) $\times$ $O(\log n)$.
Gives sparsifiers with $O(n \log n)$ edges – reducing from $O(n^2)$.

Graph Sparsification

How are sparsifiers constructed?

Sampling probabilities:

- Connectivity for cut sparsifiers [Benczúr, Karger '96], [Fung, Hariharan, Harvey, Panigrahi '11].
- **Effective resistances** (i.e statistical leverage scores) for spectral sparsifiers [Spielman, Srivastava '08].

Actually oversample: by (effective resistance) $\times$ $O(\log n)$.
Gives sparsifiers with $O(n \log n)$ edges – reducing from $O(n^2)$.

Graph Sparsification

How are sparsifiers constructed?

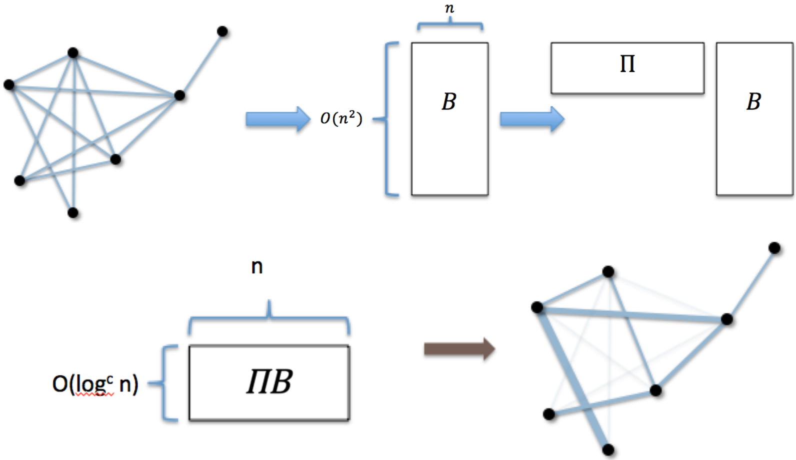
Sampling probabilities:

- Connectivity for cut sparsifiers [Benczúr, Karger '96], [Fung, Hariharan, Harvey, Panigrahi '11].
- **Effective resistances** (i.e statistical leverage scores) for spectral sparsifiers [Spielman, Srivastava '08].

Actually oversample: by (effective resistance) $\times$ $O(\log n)$.
Gives sparsifiers with $O(n \log n)$ edges – reducing from $O(n^2)$.

Application 2: Spectral Sparsification

Highest Level Idea Of Our Approach



Application 2: Spectral Sparsification

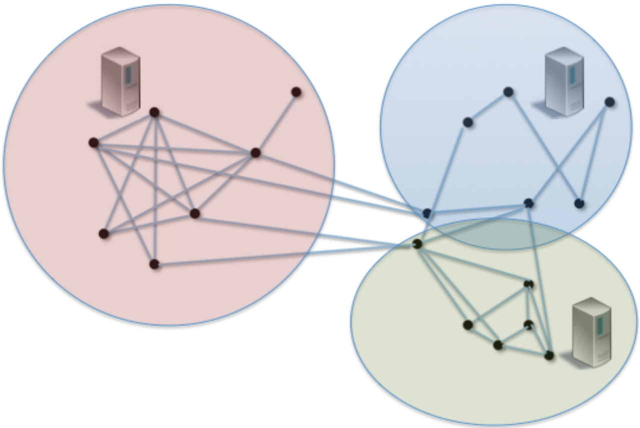
Why?

- Semi-streaming model with insertions and deletions
- Near optimal *oblivious* graph compression
- **Distributed Graph Computations**

Application 2: Spectral Sparsification

Distributed Graph Computation

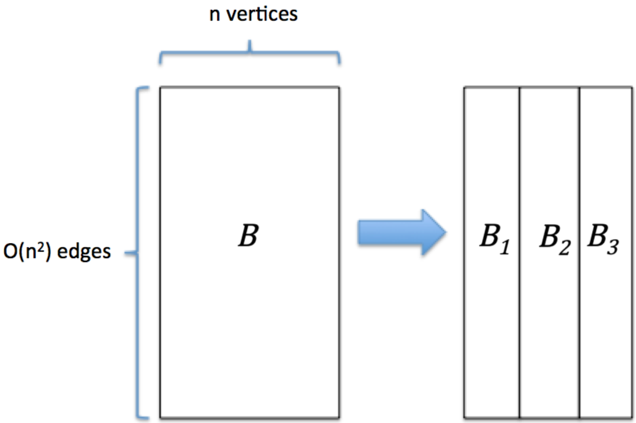
- Trinity, Pregel, Giraph



Application 2: Spectral Sparsification

Distributed Graph Computation

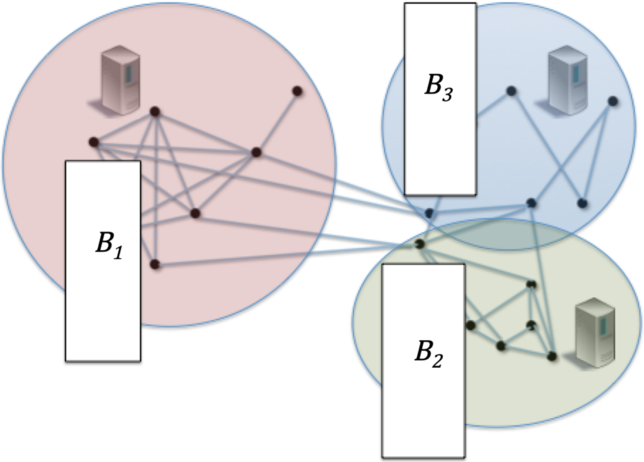
- Trinity, Pregel, Giraph



Application 2: Spectral Sparsification

Distributed Graph Computation

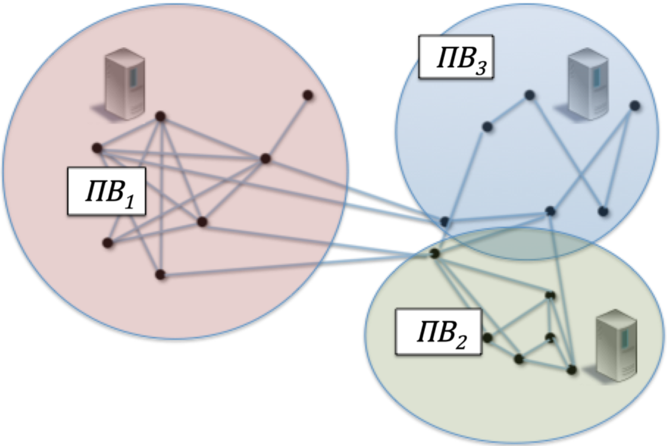
- Trinity, Pregel, Giraph



Application 2: Spectral Sparsification

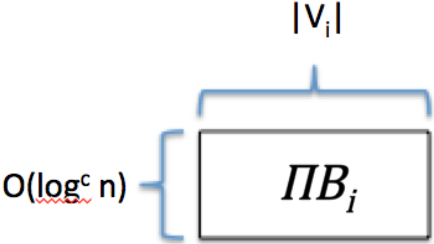
Distributed Graph Computation

- Trinity, Pregel, Giraph



Application 2: Spectral Sparsification

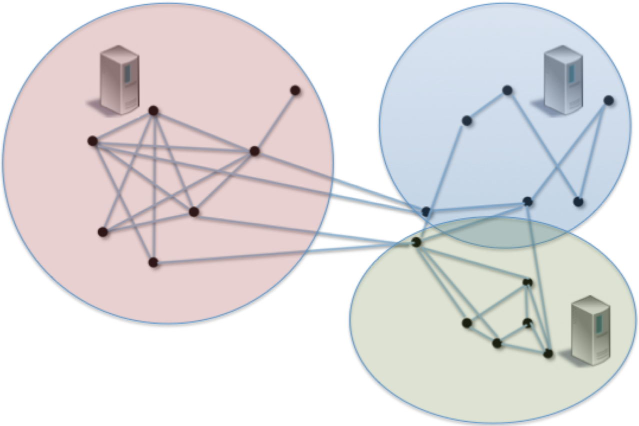
- Naive to share my data: $O(|V_i|n)$
- With sketching: $O(|V_i|\log^c n)$



Application 2: Spectral Sparsification

Alternatives to Sketching?

- Simulate message passing algorithms over the nodes - this is what's done in practice.



Application 2: Spectral Sparsification

Alternatives to Sketching?

- Koutis '14 gives distributed algorithm for spectral sparsification
- Iteratively computes $O(\log n)$ spanners (alternatively, low stretch trees) to upper bound effective resistances and sample edges.
- *Combinatorial* and *local*

Application 2: Spectral Sparsification

- Cost per spanner: $O(\log^2 n)$ rounds, $O(m \log n)$ messages, $O(\log n)$ message size.
- If simulating, each server sends $O(\delta(V_i) \log n)$ per round.
- $O(\delta(V_i) \log n)$ beats our bound of $O(|V_i| \log n)$ iff $\delta(V_i) \leq |V_i|$
- But in that case, just keep all your outgoing edges and sparsify locally! At worst adds n edges to the final sparsifier.

Application 2: Spectral Sparsification

Moral of That Story?

- I'm not sure.
- Sparsifiers are very strong. Could we do better for other problems?
- Can we reduce communication of simulated distributed protocols using sparsifiers?
- What other things can sketches be applied to? Biggest open question is distances - spanners, etc.

Sketching a Sparsifier

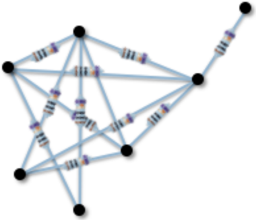
We are still going to sample by effective resistance.

- Treat graph as resistor network, each edge has resistance 1.
- Flow 1 unit of current from node i to j and measure voltage drop between the nodes.

Sketching a Sparsifier

We are still going to sample by effective resistance.

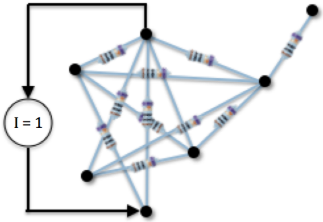
- Treat graph as resistor network, each edge has resistance 1.
- Flow 1 unit of current from node i to j and measure voltage drop between the nodes.



Sketching a Sparsifier

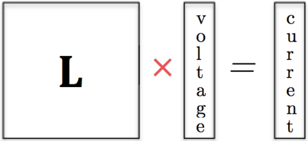
We are still going to sample by effective resistance.

- Treat graph as resistor network, each edge has resistance 1.
- Flow 1 unit of current from node i to j and measure voltage drop between the nodes.



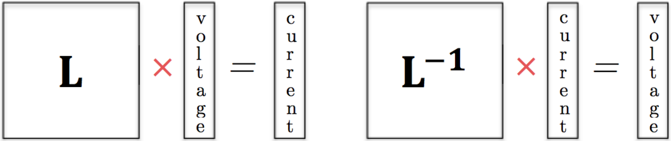
Sketching a Sparsifier

Using standard $V = IR$ equations:



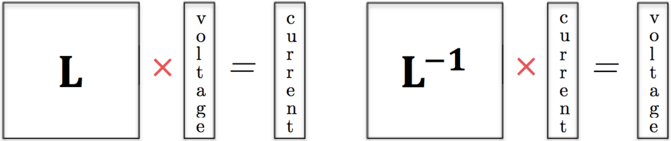
Sketching a Sparsifier

Using standard $V = IR$ equations:



Sketching a Sparsifier

Using standard $V = IR$ equations:



If $\mathbf{x}_e = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \\ 0 \end{bmatrix}$, e's effective resistance is $\tau_e = \mathbf{x}_e^\top \mathbf{L}^{-1} \mathbf{x}_e$.

Sketching a Sparsifier

Effective resistance of edge e is $\tau_e = \mathbf{x}_e^\top \mathbf{L}^{-1} \mathbf{x}_e$.

Alternatively, τ_e is the e^{th} entry in the vector:

$$\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$$

AND

$$\tau_e = \mathbf{x}_e^\top \mathbf{L}^{-1} \mathbf{x}_e = \mathbf{x}_e^\top (\mathbf{L}^{-1})^\top \mathbf{B}^\top \mathbf{B} \mathbf{L}^{-1} \mathbf{x}_e = \|\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e\|_2^2$$

Sketching a Sparsifier

Effective resistance of edge e is $\tau_e = \mathbf{x}_e^\top \mathbf{L}^{-1} \mathbf{x}_e$.

Alternatively, τ_e is the e^{th} entry in the vector:

$$\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$$

AND

$$\tau_e = \mathbf{x}_e^\top \mathbf{L}^{-1} \mathbf{x}_e = \mathbf{x}_e^\top (\mathbf{L}^{-1})^\top \mathbf{B}^\top \mathbf{B} \mathbf{L}^{-1} \mathbf{x}_e = \|\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e\|_2^2$$

Sketching a Sparsifier

Effective resistance of edge e is $\tau_e = \mathbf{x}_e^\top \mathbf{L}^{-1} \mathbf{x}_e$.

Alternatively, τ_e is the e^{th} entry in the vector:

$$\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$$

AND

$$\tau_e = \mathbf{x}_e^\top \mathbf{L}^{-1} \mathbf{x}_e = \mathbf{x}_e^\top (\mathbf{L}^{-1})^\top \mathbf{B}^\top \mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e = \|\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e\|_2^2$$

Sketching a Sparsifier

We just need two more ingredients: $\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$

ℓ_2 **Heavy Hitters [GLPS10]:**

- Sketch vector *poly*(n) vector in *polylog*(n) space.
- Extract any element who's square is a $O(1/\log n)$ fraction of the vector's squared norm.

Coarse Sparsifier:

- $\tilde{\mathbf{L}}$ such that $\mathbf{x}^\top \tilde{\mathbf{L}} \mathbf{x} = (1 \pm \mathit{constant}) \mathbf{x}^\top \mathbf{L} \mathbf{x}$

Sketching a Sparsifier

We just need two more ingredients: $\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$

ℓ_2 **Heavy Hitters [GLPS10]:**

- Sketch vector *poly*(n) vector in *polylog*(n) space.
- Extract any element who's square is a $O(1/\log n)$ fraction of the vector's squared norm.

Coarse Sparsifier:

- $\tilde{\mathbf{L}}$ such that $\mathbf{x}^\top \tilde{\mathbf{L}} \mathbf{x} = (1 \pm \textit{constant}) \mathbf{x}^\top \mathbf{L} \mathbf{x}$

Sketching a Sparsifier

We just need two more ingredients: $\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$

ℓ_2 **Heavy Hitters [GLPS10]:**

- Sketch vector *poly*(n) vector in *polylog*(n) space.
- Extract any element who's square is a $O(1/\log n)$ fraction of the vector's squared norm.

Coarse Sparsifier:

- $\tilde{\mathbf{L}}$ such that $\mathbf{x}^\top \tilde{\mathbf{L}} \mathbf{x} = (1 \pm \mathit{constant}) \mathbf{x}^\top \mathbf{L} \mathbf{x}$

Sketching a Sparsifier

Putting it all together: $\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$

1. Sketch $(\mathbf{\Pi}_{\text{heavy hitters}})\mathbf{B}$ in $n \log^c n$ space.
2. Compute $(\mathbf{\Pi}_{\text{heavy hitters}})\mathbf{B}\tilde{\mathbf{L}}^{-1}$.
3. For every possible edge e, compute $(\mathbf{\Pi}_{\text{heavy hitters}})\mathbf{B}\tilde{\mathbf{L}}^{-1}\mathbf{x}_e$
4. Extract heavy hitters from the vector, check if e^{th} entry is one.

$$\frac{\mathbf{B}\tilde{\mathbf{L}}^{-1}\mathbf{x}_e(e)^2}{\|\mathbf{B}\tilde{\mathbf{L}}^{-1}\mathbf{x}_e\|_2^2} \approx \frac{\tau_e^2}{\tau_e} = \tau_e$$

So, as long as $\tau_e > O(1/\log n)$, we will recover the edge!

Sketching a Sparsifier

Putting it all together: $\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$

1. Sketch $(\mathbf{\Pi}_{\text{heavy hitters}})\mathbf{B}$ in $n \log^c n$ space.
2. Compute $(\mathbf{\Pi}_{\text{heavy hitters}})\mathbf{B}\tilde{\mathbf{L}}^{-1}$.
3. For every possible edge e, compute $(\mathbf{\Pi}_{\text{heavy hitters}})\mathbf{B}\tilde{\mathbf{L}}^{-1}\mathbf{x}_e$
4. Extract heavy hitters from the vector, check if e^{th} entry is one.

$$\frac{\mathbf{B}\tilde{\mathbf{L}}^{-1}\mathbf{x}_e(e)^2}{\|\mathbf{B}\tilde{\mathbf{L}}^{-1}\mathbf{x}_e\|_2^2} \approx \frac{\tau_e^2}{\tau_e} = \tau_e$$

So, as long as $\tau_e > O(1/\log n)$, we will recover the edge!

Sketching a Sparsifier

Putting it all together: $\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$

1. Sketch $(\mathbf{\Pi}_{\text{heavy hitters}})\mathbf{B}$ in $n \log^c n$ space.
2. Compute $(\mathbf{\Pi}_{\text{heavy hitters}})\mathbf{B}\tilde{\mathbf{L}}^{-1}$.
3. For every possible edge e, compute $(\mathbf{\Pi}_{\text{heavy hitters}})\mathbf{B}\tilde{\mathbf{L}}^{-1}\mathbf{x}_e$
4. Extract heavy hitters from the vector, check if e^{th} entry is one.

$$\frac{\mathbf{B}\tilde{\mathbf{L}}^{-1}\mathbf{x}_e(e)^2}{\|\mathbf{B}\tilde{\mathbf{L}}^{-1}\mathbf{x}_e\|_2^2} \approx \frac{\tau_e^2}{\tau_e} = \tau_e$$

So, as long as $\tau_e > O(1/\log n)$, we will recover the edge!

Sketching a Sparsifier

Putting it all together: $\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$

1. Sketch $(\mathbf{\Pi}_{\text{heavy hitters}})\mathbf{B}$ in $n \log^c n$ space.
2. Compute $(\mathbf{\Pi}_{\text{heavy hitters}})\mathbf{B}\tilde{\mathbf{L}}^{-\mathbf{1}}$.
3. For every possible edge e, compute $(\mathbf{\Pi}_{\text{heavy hitters}})\mathbf{B}\tilde{\mathbf{L}}^{-\mathbf{1}}\mathbf{x}_e$
4. Extract heavy hitters from the vector, check if e^{th} entry is one.

$$\frac{\mathbf{B}\tilde{\mathbf{L}}^{-1}\mathbf{x}_e(e)^2}{\|\mathbf{B}\tilde{\mathbf{L}}^{-1}\mathbf{x}_e\|_2^2} \approx \frac{\tau_e^2}{\tau_e} = \tau_e$$

So, as long as $\tau_e > O(1/\log n)$, we will recover the edge!

Sketching a Sparsifier

Putting it all together: $\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$

1. Sketch $(\mathbf{\Pi}_{\text{heavy hitters}})\mathbf{B}$ in $n \log^c n$ space.
2. Compute $(\mathbf{\Pi}_{\text{heavy hitters}})\mathbf{B}\tilde{\mathbf{L}}^{-1}$.
3. For every possible edge e, compute $(\mathbf{\Pi}_{\text{heavy hitters}})\mathbf{B}\tilde{\mathbf{L}}^{-1}\mathbf{x}_e$
4. Extract heavy hitters from the vector, check if e^{th} entry is one.

$$\frac{\mathbf{B}\tilde{\mathbf{L}}^{-1}\mathbf{x}_e(e)^2}{\|\mathbf{B}\tilde{\mathbf{L}}^{-1}\mathbf{x}_e\|_2^2} \approx \frac{\tau_e^2}{\tau_e} = \tau_e$$

So, as long as $\tau_e > O(1/\log n)$, we will recover the edge!

Sketching a Sparsifier

How about edges with lower effective resistance? Sketch:

$\mathbf{BL}^{-1}\mathbf{x}_e$

Sketching a Sparsifier

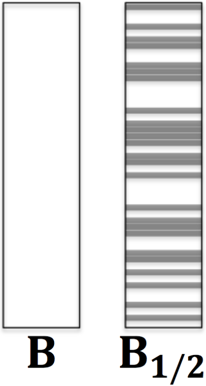
How about edges with lower effective resistance? Sketch:

$\mathbf{B}$

$\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$

Sketching a Sparsifier

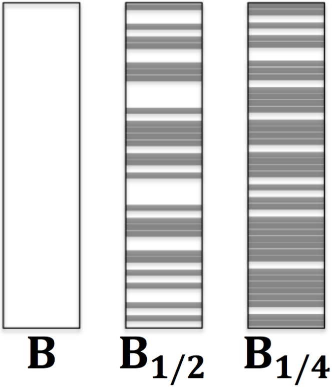
How about edges with lower effective resistance? Sketch:



$\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$

Sketching a Sparsifier

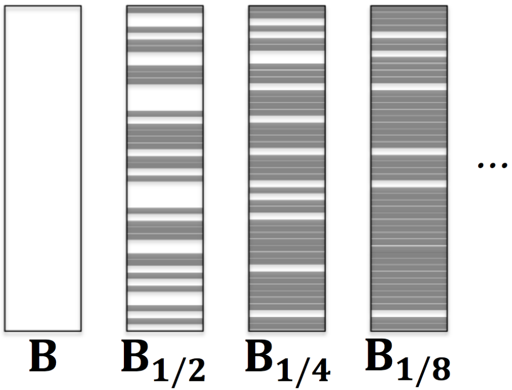
How about edges with lower effective resistance? Sketch:



$\mathbf{BL^{-1}x}_e$

Sketching a Sparsifier

How about edges with lower effective resistance? Sketch:



$\mathbf{BL^{-1}x}_e$

Sketching a Sparsifier

$\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$

How about edges with lower effective resistance?

- First level: $\tau_e > 1/\log n$ with probability 1.
- Second level: $\tau_e > 1/2\log n$ with probability 1/2.
- Third level: $\tau_e > 1/4\log n$ with probability 1/4.
- Forth level: $\tau_e > 1/8\log n$ with probability 1/8.
- ...

So, we can sample every edge by (effective resistance) $\times$ $O(\log n)$.

Sketching a Sparsifier

$\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$

How about edges with lower effective resistance?

- First level: $\tau_e > 1/\log n$ with probability 1.
- Second level: $\tau_e > 1/2 \log n$ with probability 1/2.
- Third level: $\tau_e > 1/4 \log n$ with probability 1/4.
- Forth level: $\tau_e > 1/8 \log n$ with probability 1/8.
- ...

So, we can sample every edge by (effective resistance) $\times$ $O(\log n)$.

Sketching a Sparsifier

$\mathbf{BL}^{-1}\mathbf{x}_e$

How about edges with lower effective resistance?

- First level: $\tau_e > 1/\log n$ with probability 1.
- Second level: $\tau_e > 1/2\log n$ with probability 1/2.
- Third level: $\tau_e > 1/4\log n$ with probability 1/4.
- Forth level: $\tau_e > 1/8\log n$ with probability 1/8.
- ...

So, we can sample every edge by (effective resistance) $\times$ $O(\log n)$.

Sketching a Sparsifier

$\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$

How about edges with lower effective resistance?

- First level: $\tau_e > 1/\log n$ with probability 1.
- Second level: $\tau_e > 1/2\log n$ with probability $1/2$.
- Third level: $\tau_e > 1/4\log n$ with probability $1/4$.
- Forth level: $\tau_e > 1/8\log n$ with probability $1/8$.
- ...

So, we can sample every edge by (effective resistance) $\times$ $O(\log n)$.

Sketching a Sparsifier

$\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$

How about edges with lower effective resistance?

- First level: $\tau_e > 1/\log n$ with probability 1.
- Second level: $\tau_e > 1/2\log n$ with probability $1/2$.
- Third level: $\tau_e > 1/4\log n$ with probability $1/4$.
- Forth level: $\tau_e > 1/8\log n$ with probability $1/8$.
- ...

So, we can sample every edge by (effective resistance) $\times$ $O(\log n)$.

Sketching a Sparsifier

$\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$

How about edges with lower effective resistance?

- First level: $\tau_e > 1/\log n$ with probability 1.
- Second level: $\tau_e > 1/2 \log n$ with probability $1/2$.
- Third level: $\tau_e > 1/4 \log n$ with probability $1/4$.
- Forth level: $\tau_e > 1/8 \log n$ with probability $1/8$.
- ...

So, we can sample every edge by (effective resistance) $\times$ $O(\log n)$.

Sketching a Sparsifier

$\mathbf{B}\mathbf{L}^{-1}\mathbf{x}_e$

How about edges with lower effective resistance?

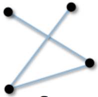
- First level: $\tau_e > 1/\log n$ with probability 1.
- Second level: $\tau_e > 1/2\log n$ with probability $1/2$.
- Third level: $\tau_e > 1/4\log n$ with probability $1/4$.
- Forth level: $\tau_e > 1/8\log n$ with probability $1/8$.
- ...

So, we can sample every edge by (effective resistance) $\times$ $O(\log n)$.

Sparsifer Chain

Final Piece [Li, Miller, Peng '12]

- We needed a constant error spectral sparsifier to get our $(1 \pm \epsilon)$ sparsifier.

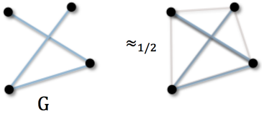


G

Sparsifer Chain

Final Piece [Li, Miller, Peng '12]

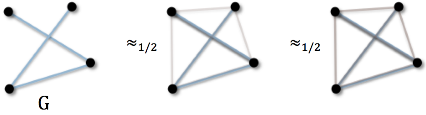
- We needed a constant error spectral sparsifier to get our $(1 \pm \epsilon)$ sparsifier.



Sparsifer Chain

Final Piece [Li, Miller, Peng '12]

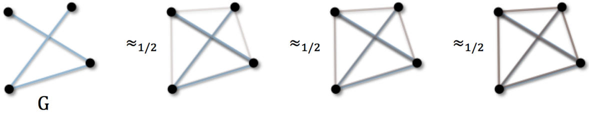
- We needed a constant error spectral sparsifier to get our $(1 \pm \epsilon)$ sparsifier.



Sparsifer Chain

Final Piece [Li, Miller, Peng '12]

- We needed a constant error spectral sparsifier to get our $(1 \pm \epsilon)$ sparsifier.



Sparsifer Chain

Final Piece [Li, Miller, Peng '12]

- We needed a constant error spectral sparsifier to get our $(1 \pm \epsilon)$ sparsifier.