Spectral Sparsification: Constructions and Applications

He Sun

University of Bristol
Graph sparsification

Why do we need graph sparsification?
- It is more space-efficient to store sparse graphs.
- Many algorithms run faster on sparse graphs.
Graph sparsification

Why do we need graph sparsification?

It is more space-efficient to store sparse graphs.
Many algorithms run faster on sparse graphs.
Why do we need graph sparsification?

- It is more space-efficient to store sparse graphs.
Why do we need graph sparsification?

- It is more space-efficient to store sparse graphs.
- Many algorithms run faster on sparse graphs.
For any undirected graph $G$ with $n$ vertices and weight $w : V \times V \to \mathbb{R}_{\geq 0}$, the Laplacian matrix of $G$ is defined by

$$L_G(u, v) = \begin{cases} -w(u, v) & \text{if } u \neq v, \\ \sum_{u \sim z} w(u, z) & \text{if } u = v. \end{cases}$$
For any undirected graph $G$ with $n$ vertices and weight $w : V \times V \to \mathbb{R}_{\geq 0}$, the Laplacian matrix of $G$ is defined by

$$L_G(u, v) = \begin{cases} -w(u, v) & \text{if } u \neq v, \\ \sum_{u \sim z} w(u, z) & \text{if } u = v. \end{cases}$$

Example:

![Graph Image]

$$L_G = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix}$$
Example: Let $S \subset V$, and define $x \in \{0, 1\}^n$ where

$$x_u = \begin{cases} 1 & \text{if } u \in S, \\ 0 & \text{otherwise}. \end{cases}$$
Spectral sparsification

Example: Let $S \subset V$, and define $x \in \{0, 1\}^n$ where

$$x_u = \begin{cases} 1 & \text{if } u \in S, \\ 0 & \text{otherwise.} \end{cases}$$

Then,

$$x^\top L_G x = \sum_{u \sim v} w(u,v) (x_u - x_v)^2 = w(S, V \setminus S).$$

For any undirected graph $G$, we call a sparse subgraph $H$ of $G$ a spectral sparsifier of $G$, if it holds for any $x \in \mathbb{R}^n$ that

$$0.9 \cdot x^\top L_H x \leq x^\top L_G x \leq 1.1 \cdot x^\top L_H x.$$
Spectral sparsification

Example: Let $S \subset V$, and define $x \in \{0, 1\}^n$ where

$$x_u = \begin{cases} 1 & \text{if } u \in S, \\ 0 & \text{otherwise.} \end{cases}$$

Then,

$$x^T L_G x = \sum_{u \sim v} w(u, v)(x_u - x_v)^2 = w(S, V \setminus S)$$

For any undirected graph $G$, we call a sparse subgraph $H$ of $G$ a spectral sparsifier of $G$, if it holds for any $x \in \mathbb{R}^n$ that

$$0.9 \cdot x^T L_H x \leq x^T L_G x \leq 1.1 \cdot x^T L_H x.$$
Example: Let $S \subset V$, and define $x \in \{0, 1\}^n$ where
\[
x_u = \begin{cases} 
1 & \text{if } u \in S, \\
0 & \text{otherwise}.
\end{cases}
\]
Then,
\[
x^T L_G x = \sum_{u \sim v} w(u, v) (x_u - x_v)^2 = w(S, V \setminus S)
\]

For any undirected graph $G$, we call a sparse subgraph $H$ of $G$ a spectral sparsifier of $G$, if it holds for any $x \in \mathbb{R}^n$ that
\[
0.9 \cdot x^T L_H x \leq x^T L_G x \leq 1.1 \cdot x^T L_H x.
\]
Spectral sparsification

Example: Let $S \subset V$, and define $x \in \{0, 1\}^n$ where

$$x_u = \begin{cases} 1 & \text{if } u \in S, \\ 0 & \text{otherwise.} \end{cases}$$

Then,

$$x^T L_G x = \sum_{u \sim v} w(u, v) (x_u - x_v)^2 = w(S, V \setminus S)$$

Spectral sparsification

For any undirected graph $G$, we call a sparse subgraph $H$ of $G$ a spectral sparsifier of $G$, if it holds for any $x \in \mathbb{R}^n$ that

$$0.9 \cdot x^T L_H x \leq x^T L_G x \leq 1.1 \cdot x^T L_H x.$$
**Spectral sparsification**

**Example:** Let $S \subset V$, and define $x \in \{0, 1\}^n$ where

$$x_u = \begin{cases} 
1 & \text{if } u \in S, \\
0 & \text{otherwise}. 
\end{cases}$$

Then,

$$x^\top L_G x = \sum_{u \sim v} w(u, v)(x_u - x_v)^2 = w(S, V \setminus S)$$

For any undirected graph $G$, we call a sparse subgraph $H$ of $G$ a spectral sparsifier of $G$, if it holds for any $x \in \mathbb{R}^n$ that

$$0.9 \cdot x^\top L_H x \leq x^\top L_G x \leq 1.1 \cdot x^\top L_H x.$$  

A spectral sparsifier preserves all cut values!
For any undirected graph $G$, we call a sparse subgraph $H$ of $G$ a spectral sparsifier of $G$, if it holds for any $x \in \mathbb{R}^n$ that

$$0.9 \cdot x^T L_H x \leq x^T L_G x \leq 1.1 \cdot x^T L_H x.$$

- All cut values between $G$ and $H$ are preserved.
For any undirected graph $G$, we call a sparse subgraph $H$ of $G$ a spectral sparsifier of $G$, if it holds for any $x \in \mathbb{R}^n$ that

$$0.9 \cdot x^T L_H x \leq x^T L_G x \leq 1.1 \cdot x^T L_H x.$$ 

- All cut values between $G$ and $H$ are preserved.
- All eigenvalues between $L_G$ and $L_H$ are preserved.
For any undirected graph $G$, we call a sparse subgraph $H$ of $G$ a spectral sparsifier of $G$, if it holds for any $x \in \mathbb{R}^n$ that

$$0.9 \cdot x^T L_H x \leq x^T L_G x \leq 1.1 \cdot x^T L_H x.$$

- All cut values between $G$ and $H$ are preserved.
- All eigenvalues between $L_G$ and $L_H$ are preserved.
For any undirected graph $G$, we call a sparse subgraph $H$ of $G$ a spectral sparsifier of $G$, if it holds for any $x \in \mathbb{R}^n$ that

$$0.9 \cdot x^T L_H x \leq x^T L_G x \leq 1.1 \cdot x^T L_H x.$$

- All cut values between $G$ and $H$ are preserved.
- All eigenvalues between $L_G$ and $L_H$ are preserved.
- Random walks in $G$ and $H$ have approximately the same mixing and cover time.
For any undirected graph $G$, we call a sparse subgraph $H$ of $G$ a spectral sparsifier of $G$, if it holds for any $x \in \mathbb{R}^n$ that

$$0.9 \cdot x^T L_H x \leq x^T L_G x \leq 1.1 \cdot x^T L_H x.$$ 

- All cut values between $G$ and $H$ are preserved.
- All eigenvalues between $L_G$ and $L_H$ are preserved.
- Random walks in $G$ and $H$ have approximately the same mixing and cover time.

**Key questions:**

- How sparse could $H$ be?
- How fast can we construct $H$?
Spectral sparsification

For any undirected graph $G$, we call a sparse subgraph $H$ of $G$ a spectral sparsifier of $G$, if it holds for any $x \in \mathbb{R}^n$ that

$$0.9 \cdot x^T L_H x \leq x^T L_G x \leq 1.1 \cdot x^T L_H x.$$ 

- All cut values between $G$ and $H$ are preserved.
- All eigenvalues between $L_G$ and $L_H$ are preserved.
- Random walks in $G$ and $H$ have approximately the same mixing and cover time.

Key questions:
- How sparse could $H$ be?
- How fast can we construct $H$?

![Algorithm's runtime vs. number of edges in $H$]
Spectral sparsification

For any undirected graph $G$, we call a sparse subgraph $H$ of $G$ a spectral sparsifier of $G$, if it holds for any $x \in \mathbb{R}^n$ that

$$0.9 \cdot x^T L_H x \leq x^T L_G x \leq 1.1 \cdot x^T L_H x.$$ 

- All cut values between $G$ and $H$ are preserved.
- All eigenvalues between $L_G$ and $L_H$ are preserved.
- Random walks in $G$ and $H$ have approximately the same mixing and cover time.

Key questions:
- How sparse could $H$ be?
- How fast can we construct $H$?
Progress on constructing spectral sparsifiers

Spielman-Teng, 2004

For any undirected graph $G$, there is a spectral sparsifier of $G$ with $O(n \text{ poly log } n)$ edges that can be constructed in $O(m \text{ poly log } n)$ time.

Algorithm's runtime

$\Omega(n)$

$\Omega(m)$

number of edges in $H$
Progress on constructing spectral sparsifiers

Spielman-Teng, 2004

For any undirected graph $G$, there is a spectral sparsifier of $G$ with $O(n \text{ poly log } n)$ edges that can be constructed in $O(m \text{ poly log } n)$ time.

Note: A constant-degree expander with $O(n)$ edges is a spectral sparsifier of a clique!
Progress on constructing spectral sparsifiers

For any undirected graph $G$, there is a spectral sparsifier of $G$ with $O(n \text{ poly log } n)$ edges that can be constructed in $O(m \text{ poly log } n)$ time.

Note: A constant-degree expander with $O(n)$ edges is a spectral sparsifier of a clique!
Progress on constructing spectral sparsifiers

Spielman-Teng, 2004

For any undirected graph $G$, there is a spectral sparsifier of $G$ with $O(n \text{ poly log } n)$ edges that can be constructed in $O(m \text{ poly log } n)$ time.

Algorithm's runtime

$O(n^6)$

$O(m \text{ poly log } n)$

$\Omega(m)$

$\Omega(n)$

$O(n \text{ poly log } n)$

number of edges in $H$

Note: A constant-degree expander with $O(n)$ edges is a spectral sparsifier of a clique!
Progress on constructing spectral sparsifiers

For any undirected graph $G$, there is a spectral sparsifier of $G$ with $O(n \text{ poly log } n)$ edges that can be constructed in $O(m \text{ poly log } n)$ time.

Algorithm's runtime

- $O(n^6)$
- $O(m^{2.1})$
- $O(m \text{ poly log } n)$
- $\Omega(m)$

number of edges in $H$

- $\Omega(n)$
- $O(n \text{ poly log } n)$

Note: A constant-degree expander with $O(n)$ edges is a spectral sparsifier of a clique!
Progress on constructing spectral sparsifiers

For any undirected graph $G$, there is a spectral sparsifier of $G$ with $O(n \text{ poly log } n)$ edges that can be constructed in $O(m \text{ poly log } n)$ time.

Spielman-Teng, 2004

Algorithm’s runtime

- $O(n^6)$
- $O(m^{2.1})$
- $O(m \text{ poly log } n)$
- $\Omega(m)$

Number of edges in $H$

- $\Omega(n)$
- $O(n \text{ poly log } n)$

Note: A constant-degree expander with $O(n)$ edges is a spectral sparsifier of a clique!
Equivalent definition of spectral sparsification

For every edge \( e = \{u, v\} \), we define

\[
b_e = (0, \ldots, 0, 1, 0, \ldots 0, -1, 0, \ldots, 0)^T
\]

\( u \)th coordinate \( v \)th coordinate

Then we can write

\[
L_G = \sum_{e \in E} w(e) b_e b_e^T
\]

Given a graph \( G \) with the Laplacian matrix \( L_G = \sum_{e \in E} w(e) b_e b_e^T \), find coefficients \( \{c_e\} \) with \( O(n) \) non-zeros, such that

\[
L_G \approx L_H = \sum_{e \in E} c_e b_e b_e^T
\]

Spectral sparsification for graphs

Given \( m \) vectors \( v_1, \ldots, v_m \) that satisfy

\[
I = \sum_{i} v_i v_i^T
\]

find coefficients \( \{c_i\}_{i=1}^m \) with \( O(n) \) non-zeros, such that

\[
I \approx \sum_{i} c_i v_i v_i^T
\]

Matrix sparsification
Equivalent definition of spectral sparsification

For every edge $e = \{u, v\}$, we define

$$b_e = (0, \ldots, 0, 1, 0, \ldots 0, -1, 0, \ldots, 0)^T$$

Then we can write

$$L_G = \sum_{e=\{u,v\} \in E} w(u, v) b_e b_e^T$$
Equivalent definition of spectral sparsification

For every edge $e = \{u, v\}$, we define

$$b_e = (0, \ldots, 0, 1, 0, \ldots, 0, -1, 0, \ldots, 0)^T$$

Then we can write

$$L_G = \sum_{e=\{u,v\} \in E} w(u,v) b_e b_e^T$$

Spectral sparsification for graphs

Given a graph $G$ with the Laplacian matrix

$$L_G = \sum_{e \in E} w_e b_e b_e^T,$$

find coefficients $\{c_e\}$ with $O(n)$ non-zeros, such that

$$L_G \approx L_H = \sum_{e \in E} c_e b_e b_e^T.$$
Equivalent definition of spectral sparsification

For every edge $e = \{u, v\}$, we define

$$b_e = (0, \ldots, 0, 1, 0, \ldots, -1, 0, \ldots , 0)^T$$

Then we can write

$$L_G = \sum_{e = \{u, v\} \in E} w(u, v) b_e b_e^T$$

Spectral sparsification for graphs

Given a graph $G$ with the Laplacian matrix

$$L_G = \sum_{e \in E} w_e b_e b_e^T,$$

find coefficients $\{c_e\}$ with $O(n)$ non-zeros, such that

$$L_G \approx L_H = \sum_{e \in E} c_e b_e b_e^T.$$
Any positive definite matrix $A$ defines an ellipsoid

$$\text{ellip}(A) = \{ x \in \mathbb{R}^n : x^T A^{-1} x \leq 1 \}.$$
Any positive definite matrix $A$ defines an ellipsoid

$$\text{ellip}(A) = \{ x \in \mathbb{R}^n : x^T A^{-1} x \leq 1 \}.$$ 

- Eigenvectors of $A$ define $n$ orthogonal directions of $\text{ellip}(A)$;
Any positive definite matrix $A$ defines an ellipsoid

$$\text{ellip}(A) = \{x \in \mathbb{R}^n : x^T A^{-1} x \leq 1\}.$$  

- Eigenvectors of $A$ define $n$ orthogonal directions of $\text{ellip}(A)$;
- The semi-length distances along the $i$th direction is $1/\sqrt{\lambda_i}$.  

Geometric interpretation of spectral sparsification:
Choose and re-weight $O(n)$ vectors, such that the corresponding ellipsoid is close to be a sphere.
Any positive definite matrix $A$ defines an ellipsoid

$$\text{ellip}(A) = \{ x \in \mathbb{R}^n : x^\top A^{-1} x \leq 1 \}.$$ 

- Eigenvectors of $A$ define $n$ orthogonal directions of $\text{ellip}(A)$;
- The semi-length distances along the $i$th direction is $1/\sqrt{\lambda_i}$.

$\text{ellip}(A)$ is close to be a sphere iff $A \approx c \cdot I$. 
Any positive definite matrix $A$ defines an ellipsoid

$$\text{ellip}(A) = \{ x \in \mathbb{R}^n : x^\top A^{-1} x \leq 1 \}.$$ 

- Eigenvectors of $A$ define $n$ orthogonal directions of $\text{ellip}(A)$;
- The semi-length distances along the $i$th direction is $1/\sqrt{\lambda_i}$.

$\text{ellip}(A)$ is close to be a sphere iff $A \approx c \cdot I$.

**Geometric interpretation of spectral sparsification**: Choose and re-weight $O(n)$ vectors, such that the corresponding ellipsoid is close to be a sphere.
Overview of our approach

General approach to construct a linear-sized spectral sparsifier

- The algorithm proceeds by iterations, and maintains two spheres $\ell_j \cdot I$ and $u_j \cdot I$ in each iteration $j$;

Overview of our approach

General approach to construct a linear-sized spectral sparsifier

- The algorithm proceeds by iterations, and maintains two spheres $\ell_j \cdot I$ and $u_j \cdot I$ in each iteration $j$;

Overview of our approach

The algorithm proceeds by iterations, and maintains two spheres $\ell_j \cdot I$ and $u_j \cdot I$ in each iteration $j$;

- The constructed ellipsoid $A_j$ always satisfies $\ell_j \cdot I \prec A_j \prec u_j \cdot I$;

Overview of our approach

General approach to construct a linear-sized spectral sparsifier

- The algorithm proceeds by iterations, and maintains two spheres \( \ell_j \cdot I \) and \( u_j \cdot I \) in each iteration \( j \);
- The constructed ellipsoid \( A_j \) always satisfies \( \ell_j \cdot I < A_j < u_j \cdot I \);

Overview of our approach

General approach to construct a linear-sized spectral sparsifier

- The algorithm proceeds by iterations, and maintains two spheres $\ell_j \cdot I$ and $u_j \cdot I$ in each iteration $j$;
- The constructed ellipsoid $A_j$ always satisfies $\ell_j \cdot I \prec A_j \prec u_j \cdot I$;

Overview of our approach

General approach to construct a linear-sized spectral sparsifier

- The algorithm proceeds by iterations, and maintains two spheres $\ell_j \cdot I$ and $u_j \cdot I$ in each iteration $j$;
- The constructed ellipsoid $A_j$ always satisfies $\ell_j \cdot I \prec A_j \prec u_j \cdot I$;
- After $T$ iterations, $\ell_T \approx u_T$ implies that $A_T \approx I$.

Overview of our approach

The algorithm proceeds by iterations, and maintains two spheres $\ell_j \cdot I$ and $u_j \cdot I$ in each iteration $j$;

- The constructed ellipsoid $A_j$ always satisfies $\ell_j \cdot I \prec A_j \prec u_j \cdot I$;
- After $T$ iterations, $\ell_T \approx u_T$ implies that $A_T \approx I$.

Overview of our approach

General approach to construct a linear-sized spectral sparsifier

- The algorithm proceeds by iterations, and maintains two spheres $\ell_j \cdot I$ and $u_j \cdot I$ in each iteration $j$;
- The constructed ellipsoid $A_j$ always satisfies $\ell_j \cdot I \prec A_j \prec u_j \cdot I$;
- After $T$ iterations, $\ell_T \approx u_T$ implies that $A_T \approx I$.

Key issues of the approach

Iteration $j$ → Iteration $j + 1$ → Final iteration $T$

Q: Control the shape of ellipsoid
A: by potential function $\Phi_{u,\ell}(A) = \text{tr} \exp(uI - A) - 1 + \text{tr} \exp(A - \ell I) - 1$

Bounded $\Phi_{u,\ell}(A)$ ensures $\ell I \prec A \prec uI$

Q: Choose a correct set of vectors in each iteration
A: Solve a specific SDP in $O(m \cdot \text{poly log } n)$ time

Q: Bound the number of iterations
A: $T = O(\text{poly log } n)$ iterations suffice
Key issues of the approach

Q: Control the shape of ellipsoid $A$

A: by potential function $\Phi(u, \ell)(A) = \text{tr} \exp(uI - A) - 1 + \text{tr} \exp(A - \ell I) - 1$

Bounded $\Phi(u, \ell)(A)$ ensures $\ell I \prec A \prec uI$

Q: Choose a correct set of vectors in each iteration

A: Solve a specific SDP in $O(m \cdot \text{poly log } n)$ time

Q: Bound the number of iterations

$T_A: T = O(\text{poly log } n)$ iterations suffice
Q: Control the shape of ellipsoid $A$

$\Phi_{u, \ell}(A) = \text{tr} \exp(uI - A) - 1 + \text{tr} \exp(A - \ell I) - 1$ is bounded.

Q: Choose a correct set of vectors in each iteration

$Q: \text{Control the shape of ellipsoid } A$

$Q: \text{Choose a correct set of vectors in each iteration}$
Key issues of the approach

Q: Control the shape of ellipsoid $A$

Q: Choose a correct set of vectors in each iteration

Q: Bound the number of iterations $T$
Key issues of the approach

Q: Control the shape of ellipsoid $A$
A: by potential function $\Phi_{u, \ell}(A) = \text{tr} \exp(uI - A)^{-1} + \text{tr} \exp(A - \ell I)^{-1}$

Q: Choose a correct set of vectors in each iteration

Q: Bound the number of iterations $T$
Key issues of the approach

Q: Control the shape of ellipsoid $A$
A: by potential function $\Phi_{u, \ell}(A) = \text{tr} \exp(uI - A)^{-1} + \text{tr} \exp(A - \ell I)^{-1}$
Bounded $\Phi_{u, \ell}(A)$ ensures $\ell I \prec A \prec uI$

Q: Choose a correct set of vectors in each iteration

Q: Bound the number of iterations $T$
Key issues of the approach

**Q: Control the shape of ellipsoid $A$**

**A:** by potential function $\Phi_{u,\ell}(A) = \text{tr} \exp(uI - A)^{-1} + \text{tr} \exp(A - \ell I)^{-1}$

Bounded $\Phi_{u,\ell}(A)$ ensures $\ell I \prec A \prec uI$

**Q: Choose a correct set of vectors in each iteration**

**A:** Solve a specific SDP in $O(m \cdot \text{poly log } n)$ time

**Q: Bound the number of iterations $T$**
**Key issues of the approach**

**Q: Control the shape of ellipsoid $A$**

**A:** by potential function $\Phi_{u, \ell}(A) = \text{tr} \exp(uI - A)^{-1} + \text{tr} \exp(A - \ell I)^{-1}$

Bounded $\Phi_{u, \ell}(A)$ ensures $\ell I \prec A \prec uI$

**Q: Choose a correct set of vectors in each iteration**

**A:** Solve a specific SDP in $O(m \cdot \text{poly log } n)$ time

**Q: Bound the number of iterations $T$**

$A : T = O(\text{poly log } n)$ iterations suffice
Algorithm for constructing a linear-sized sparsifier

1. $j = 0$, set the initial matrix $A = 0$;
2. $\ell = -1/4$, $u = 1/4$;
Our algorithm

Algorithm for constructing a linear-sized sparsifier

1: \( j = 0 \), set the initial matrix \( A = 0 \);
2: \( \ell = -1/4 \), \( u = 1/4 \);
3: \textbf{while} \( u - \ell < 1 \)
Our algorithm

Algorithm for constructing a linear-sized sparsifier

1: \( j = 0 \), set the initial matrix \( A = 0 \);
2: \( \ell = -1/4, u = 1/4 \);
3: while \( u - \ell < 1 \) 
4: Choose vectors \( v_1', \ldots, v_{\ell}' \) by solving an SDP
Our algorithm

Algorithm for constructing a linear-sized sparsifier

1: \( j = 0 \), set the initial matrix \( A = 0 \);
2: \( \ell = -1/4, \ u = 1/4 \);
3: \textbf{while} \( u - \ell < 1 \)
4: \hspace{1em} Choose vectors \( v_1', \ldots, v_\ell' \) by solving an SDP
5: \hspace{1em} \( \Delta = \sum_{i=1}^{\ell} v_i'(v_i')^\top \)
Our algorithm

Algorithm for constructing a linear-sized sparsifier

1: \( j = 0 \), set the initial matrix \( A = 0 \);
2: \( \ell = -1/4 \), \( u = 1/4 \);
3: while \( u - \ell < 1 \)
4: Choose vectors \( v'_1, \ldots, v'_\ell \) by solving an SDP
5: \( \Delta = \sum_{i=1}^{\ell} v'_i (v'_i)^\top \)
6: \( A = A + \Delta \)
Our algorithm

Algorithm for constructing a linear-sized sparsifier

1. $j = 0$, set the initial matrix $A = 0$;
2. $\ell = -1/4$, $u = 1/4$;
3. while $u - \ell < 1$
4. Choose vectors $v'_1, \ldots, v'_\ell$ by solving an SDP
5. $\Delta = \sum_{i=1}^{\ell} v'_i (v'_i)^\top$
6. $A = A + \Delta$
7. increase the value of $u$ and $\ell$
Our algorithm

Algorithm for constructing a linear-sized sparsifier

1: \( j = 0 \), set the initial matrix \( A = 0 \);
2: \( \ell = -1/4, \; u = 1/4; \)
3: **while** \( u - \ell < 1 \)
4: \hspace{1em} Choose vectors \( v'_1, \ldots, v'_\ell \) by solving an SDP
5: \hspace{1em} \[ \Delta = \sum_{i=1}^{\ell} v'_i (v'_i)^\top \]
6: \hspace{1em} \( A = A + \Delta \)
7: \hspace{1em} increase the value of \( u \) and \( \ell \)
8: **return** \( A \)
Our algorithm

Algorithm for constructing a linear-sized sparsifier

1: $j = 0$, set the initial matrix $A = 0$
2: $\ell = -1/4$, $u = 1/4$
3: while $u - \ell < 1$
4: Choose vectors $v_1', \ldots, v_\ell'$ by solving an SDP
5: $\Delta = \sum_{i=1}^{\ell} v_i'(v_i')^T$
6: $A = A + \Delta$
7: increase the value of $u$ and $\ell$
8: return $A$

Lee-S., STOC’17

A linear-sized spectral sparsifier can be constructed in nearly-linear time.
Application of spectral sparsification in clustering

Applications in clustering:
Applications in clustering:

- Distributed clustering: The dataset is allocated among remote sites.
Application of spectral sparsification in clustering

Applications in clustering:
Application of spectral sparsification in clustering

Applications in clustering:

Distributed clustering: The dataset is allocated among $s$ remote sites.
Application of spectral sparsification in clustering

Setup: Edges of graph $G$ are allocated at $s$ sites in an arbitrary way.

Objective: Design a communication-efficient algorithm for clustering.
Application of spectral sparsification in clustering

Setup: Edges of graph $G$ are allocated at $s$ sites in an arbitrary way.
Objective: Design a communication-efficient algorithm for clustering.

A naive approach:

- Every site sends all the maintained edges to the host;
- The host runs a clustering algorithm;
- Communication cost $= \Theta(m \log^c n)$ bits.
Application of spectral sparsification in clustering

Setup: Edges of graph $G$ are allocated at $s$ sites in an arbitrary way.
Objective: Design a communication-efficient algorithm for clustering.

A naive approach:
- Every site sends all the maintained edges to the host;
- The host runs a clustering algorithm;
- Communication cost $= \Theta(m \log^c n)$ bits.

Our proposed approach:
- Every site sends a spectral sparsifier of the subgraph it maintains to the host;
- The host runs a clustering algorithm;
- Communication cost $= \Theta(ns \log^c n)$ bits.
Distributed clustering based on spectral sparsification

**Lower bound:** Any algorithm with \( o(n^s) \) bits of communication cannot recover a constant fraction of a single cluster.  
[Chen-S.-Woodruff-Zhang, NIPS’16]
Distributed clustering based on spectral sparsification

**Lower bound:** Any algorithm with $o(ns)$ bits of communication cannot recover a constant fraction of a single cluster. [Chen-S.-Woodruff-Zhang, NIPS’16]

- Our proposed algorithm based on sparsification is communication optimal.
- Approx. ratio of our algorithm is the same as the best one in the centralised setting.

---

Original data; a corresponding graph has 70 million edges.

Clustering result in a centralised setting

Output of our algorithm with 6% of the edges communicated

Thank you!
Distributed clustering based on spectral sparsification

**Lower bound:** Any algorithm with $o(ns)$ bits of communication cannot recover a constant fraction of a single cluster. [Chen-S.-Woodruff-Zhang, NIPS’16]

- Our proposed algorithm based on sparsification is communication optimal.
- Approx. ratio of our algorithm is the same as the best one in the centralised setting.

Original data; a corresponding graph has 70 million edges.
Distributed clustering based on spectral sparsification

**Lower bound:** Any algorithm with $o(ns)$ bits of communication cannot recover a constant fraction of a single cluster.  
[Chen-S.-Woodruff-Zhang, NIPS’16]

- Our proposed algorithm based on sparsification is communication optimal.
- Approx. ratio of our algorithm is the same as the best one in the centralised setting.

Original data; a corresponding graph has 70 million edges.

Clustering result in a centralised setting
Distributed clustering based on spectral sparsification

**Lower bound:** Any algorithm with $o(n^s)$ bits of communication cannot recover a constant fraction of a single cluster. [Chen-S.-Woodruff-Zhang, NIPS’16]

- Our proposed algorithm based on sparsification is communication optimal.
- Approx. ratio of our algorithm is the same as the best one in the centralised setting.

Original data; a corresponding graph has 70 million edges.

Clustering result in a centralised setting

Output of our algorithm with 6% of the edges communicated
Distributed clustering based on spectral sparsification

**Lower bound:** Any algorithm with $o(ns)$ bits of communication cannot recover a constant fraction of a single cluster. [Chen-S.-Woodruff-Zhang, NIPS’16]

- Our proposed algorithm based on sparsification is communication optimal.
- Approx. ratio of our algorithm is the same as the best one in the centralised setting.

Original data; a corresponding graph has 70 million edges.

Clustering result in a centralised setting

Output of our algorithm with 6% of the edges communicated

Thank you!