Spectral Graph Sparsification: overview of theory and practical methods

Yiannis Koutis

University of Puerto Rico - Rio Piedras

National Science Foundation
WHERE DISCOVERIES BEGIN

Graph Sparsification or 'Sketching'

Compute a smaller graph that preserves some **crucial** property of the input

Motivation: Computational efficiency with approximation guarantees

- **BFS**: Breadth First Spanning Tree
- **Spanner:** Spanning subgraph that approximately preserves distances

Spectral Graph Sparsification

Compute a smaller graph that preserves some **crucial** property of the input

We want to approximately preserve the eigenvalues and eigenvectors of the graph **Laplacian**

Motivation: Speed-up many clustering and partitioning algorithms based on computing Laplacian eigenvectors

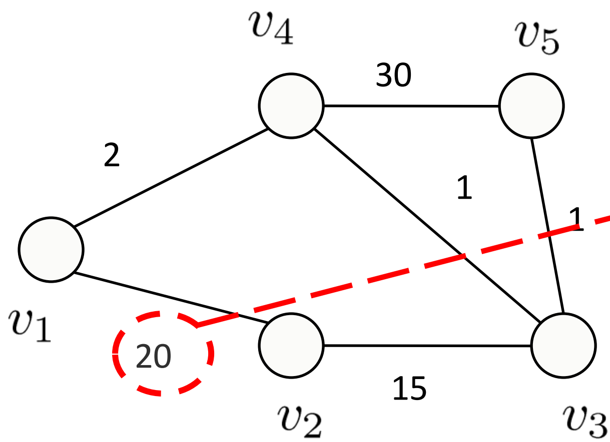
Spectral Graph Sparsification

Compute a smaller graph that preserves some **crucial** property of the input

We want to approximately preserve the **quadratic form** $\mathbf{x}^T\mathbf{L}\mathbf{x}$ of the Laplacian $\mathbf{L}$

Implies spectral approximations for both the Laplacian and the normalized Laplacian

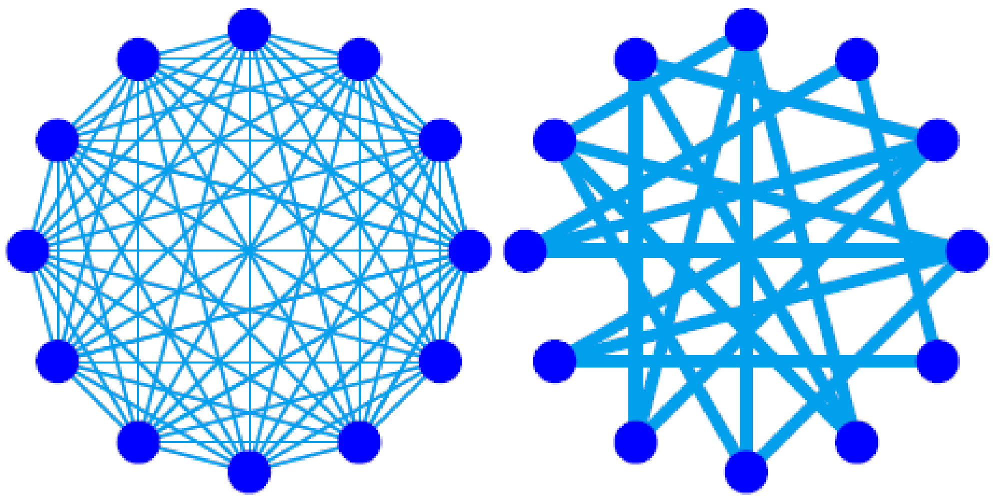
The Graph Laplacian



$$L = \begin{pmatrix} 22 & -20 & 0 & -2 & 0 \\ -20 & 35 & -15 & 0 & 0 \\ 0 & -15 & 17 & -1 & -1 \\ -2 & 0 & -1 & 33 & -30 \\ 0 & 0 & -1 & -30 & 31 \end{pmatrix}$$

Spectral Sparsification by Picture

$x^T L_G x$ $\quad$ $x^T L_H x$



- H is a reweighted subgraph of G
- H is obtained using randomness (sampling)

Combinatorial sketching

Spectral sketching

Linear system solvers

Better combinatorial sketching

Outline

- Sums of random positive matrices
- Combinatorial sketching to incremental sparsification
- Incremental sparsification for solving
- Parallel and distributed sparsification
- Deep sparsification by effective resistances
- A heuristic for better clustering

Matrix Ordering

- Whenever for all vectors *x* we have

$$x^T G x \leq x^T H x$$

- We write

$$G \preceq H$$

- In this notation, a spectral sparsifier *H* satisfies

$$(1-\epsilon)G \preceq H \preceq (1+\epsilon)G$$

Sums of random matrices

[Tropp '12, adapted]:

1. Let **S** be a nxn PSD matrix and $\mathbf{Y_1, \dots, Y_k}$ be independent random PSD matrices also of size nxn.
2. Let $\mathbf{Y = S + \sum_i Y_i}$
3. Let **Z = E[Y]**
4. Suppose $\mathbf{Y_i} \preceq \mathbf{R \cdot Z}$

S : Combinatorial Sketch

$\mathbf{Y_i}$ **: Edges**

$$
\begin{aligned}
Pr\left[Y \preceq (1-\epsilon)Z\right] &\leq n \cdot exp(-\epsilon^2/2R) \;\; \forall \epsilon \in [0,1] \\
Pr\left[Y \succeq (1+\epsilon)Z\right] &\leq n \cdot exp(-\epsilon^2/3R) \;\; \forall \epsilon \in [0,1]
\end{aligned}
$$

R: should be O(C/log n)

Outline

- Sums of random positive matrices
- Combinatorial sketching to incremental sparsification
- Incremental sparsification for solving
- Parallel and distributed sparsification
- Deep sparsification by effective resistances
- A heuristic for better clustering

A simple algorithm

1. Compute a **spanner** S':
2. Let H:= S'
3. For every edge e not in H:

   H:=H + k*e, with probability 1/k

H has O(nlog n) + m/k edges

n: number of vertices
m: number of edges

$$(1/2)Ck\log^2 n \cdot G \preceq H \preceq (3/2)Ck\log^2 n \cdot G$$

and a simple proof

- Suppose the graph G has n vertices and m edges
- For simplicity we let G be unweighted (generalization is easy)

- By definition of spanner, for every edge e of G, there is a path p_e in S ' that joins the two endpoints of e and has length logn.

- Algebraically, if G_e is the Laplacian of edge e:

$$G_e \preceq \log n \cdot p_e \preceq \log n \cdot S'$$

and a simple proof

- Apply Tropp's Theorem on: $G' = G + kS$
- Combinatorial Sketch: $S = kS'$
- Samples: $Y_i \preceq kG_e$

$$G_e \preceq \log n \cdot S \implies$$

$$Y_i \preceq kG_e \preceq \frac{1}{C \log n} kC \log^2 n \cdot S \preceq \frac{1}{C \log n} G'$$

Incremental Sparsification for Solving

H has **O(nlog n) + m/k edges**

$$(1/2)Ck\log^2 n \cdot G \preceq H \preceq (3/2)Ck\log^2 n \cdot G$$

[Spielman and Teng]

If we can construct H with same guarantees but only n+m/k edges then we can solve linear systems on Laplacians in O(mlog^{2}n) time

[K, Miller Peng 10]

Use low-stretch tree instead of spanner. It preserves distances **on average**. Use sampling with skewed probability distribution.

Incremental Sparsification for Solving

[K, Miller Peng 10,11]

If A is a symmetric diagonally dominant matrix then an approximate solution to **Ax= b** can be computed in O(m log n log log n log (1/ϵ)) time, where ϵ is the required precision

Fact:

Approximations to the j first eigenvectors can be computed via solving O(j log n) linear systems

Outline

- Sums of random positive matrices
- Combinatorial sketching to incremental sparsification
- Incremental sparsification for solving
- Parallel and distributed sparsification
- Deep sparsification by effective resistances
- A heuristic for better clustering

Parallel and Distributed Sparsification

1. Can we do better than incremental sparsification ?
2. Is there a parallel sparsification algorithm?
3. Is there a distributed sparsification algorithm ?

- Spanners hold the key to questions #2, #3
- There are **very efficient** parallel and distributed algorithms for computing spanners. From this we get parallel and distributed incremental sparsification. This doesn't itself imply parallel solvers.
- The main problem is question #1.

Parallel and Distributed Sparsification

- A better combinatorial sketch:
- **t-bundle spanner:** A collection of graphs $S_1 \ldots . S_t$ such that S_i is a spanner for $G - (S_1 + \ldots . + S_{i-1})$

- A t-bundle spanner can be computed with t sequential calls to a spanner computation algorithm.

- If t is small then the algorithm remains efficient

(polylogarithmic parallel time and distributed rounds)

A simple algorithm

1. Compute $O(\log^4 n)$-bundle spanner S
2. Let H:= S
3. For every edge e not in H:

   H:=H + 2*e, with probability 1/2

H has $O(n\log^5 n) + m/2$ edges

n: number of vertices
m: number of edges

$$(1 - 1/\log n)G \preceq H \preceq (1 + 1/\log n)G$$

A simple algorithm

- H has $O(n\log^5 n) + m/2$ edges

$$(1 - 1/\log n)H \preceq G \preceq (1 + 1/\log n)H$$

- Small size reduction (factor of 2)
- Very tight spectral approximation

- Repeat recursively on H. In $O(\log n)$ rounds we get $O(n\log^6 n)$ edges and a constant spectral approximation.

$$1/2G \preceq H \preceq 3/2G$$

A parallel solver

- This is the best known parallel sparsification routine.
- Improves the total work guarantees of a parallel solver recently described by Peng and Spielman.

- Parallel solver that works in polylogarithmic time and does **$O(m\log^3 n)$** work.

Outline

- Sums of random positive matrices
- Combinatorial sketching to incremental sparsification
- Incremental sparsification for solving
- Parallel and distributed sparsification
- Deep sparsification by effective resistances
- A heuristic for better clustering

Spielman-Srivastava: Deep sparsification

- Spielman and Srivastava proved: There is a graph H with $O(n \log n/\epsilon^2)$ edges such that
$$(1-\epsilon)G \preceq H \preceq (1+\epsilon)G$$

- The algorithm is based on sampling edges with probabilities proportional to the **effective resistances** of the edges in the graph

- Proof uses similarly Tropp's theorem

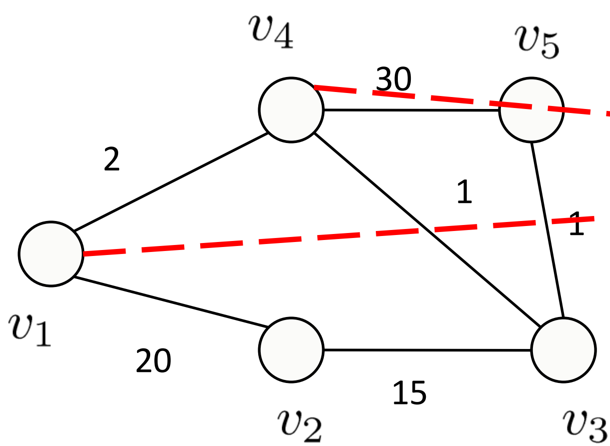
Combinatorial sketching

Spectral sketching

Linear system solvers

Better combinatorial sketching

The Incidence Matrix



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

W is the diagonal matrix containing the square roots of edge weights

Spielman-Srivastava: Deep sparsification

- **Effective resistances** can be approximated closely by **solving** O(log n) linear Laplacian systems as follows:

1. Let L be the Laplacian of the graph
2. Let B be the incident matrix of the graph
3. Let Q be a random Johnson-Lindenstrauss projection of size m x O(log n)
4. Solve the systems $L X = B^T Q$
5. Effective resistance between vertices i and j is equal to the $||X_i - X_j||_2$ where X_i is the ith row of X

- The solution X is a n x O(log n) matrix.
- Each row can be interpreted as an **embedding** of corresponding vertex to the O(log n)-dimensional Euclidean space.
- Let's call this the **effective resistance embedding**.

Fast Effective Resistances

An implementation of the Spielman-Srivastava algorithm for the quick computation of **many** effective resistances in an electrical resistive network. Effective resistances are equivalent to **commute times** of the random walk in the corresponding graph.

The code runs in MATLAB. Authored by Richard Garcia Lebron.

Download **Dependence:** CMG solver

- http://ccom.uprrp.edu/~ikoutis/SpectralAlgorithms.htm

Heuristic: clustering based on the effective resistance embedding

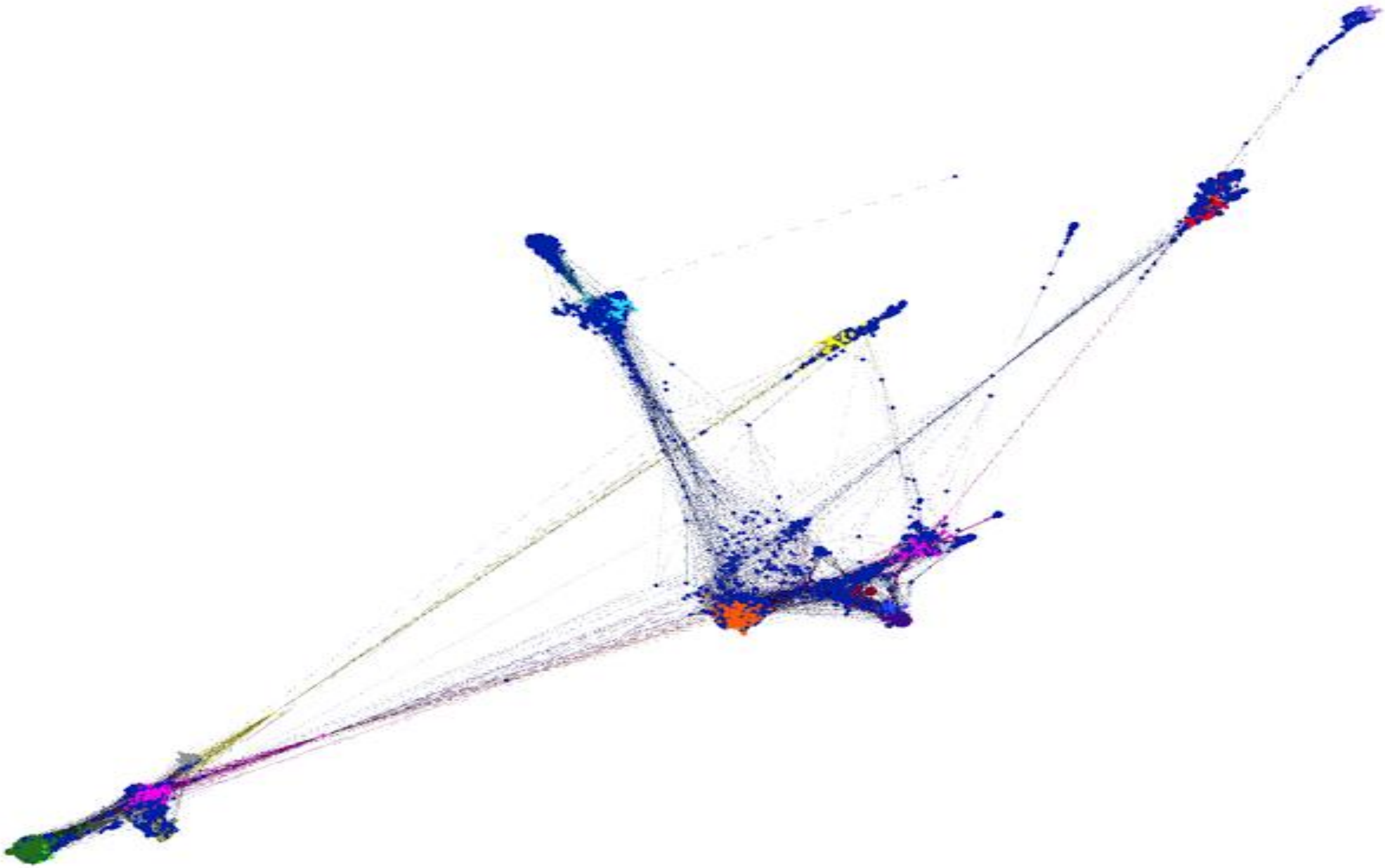
- Small effective resistance for an edge e means that there are a lot of short connections between the two endpoints .

- Points that are close in the geometric embedding should be close in this connectivity sense in the graph.

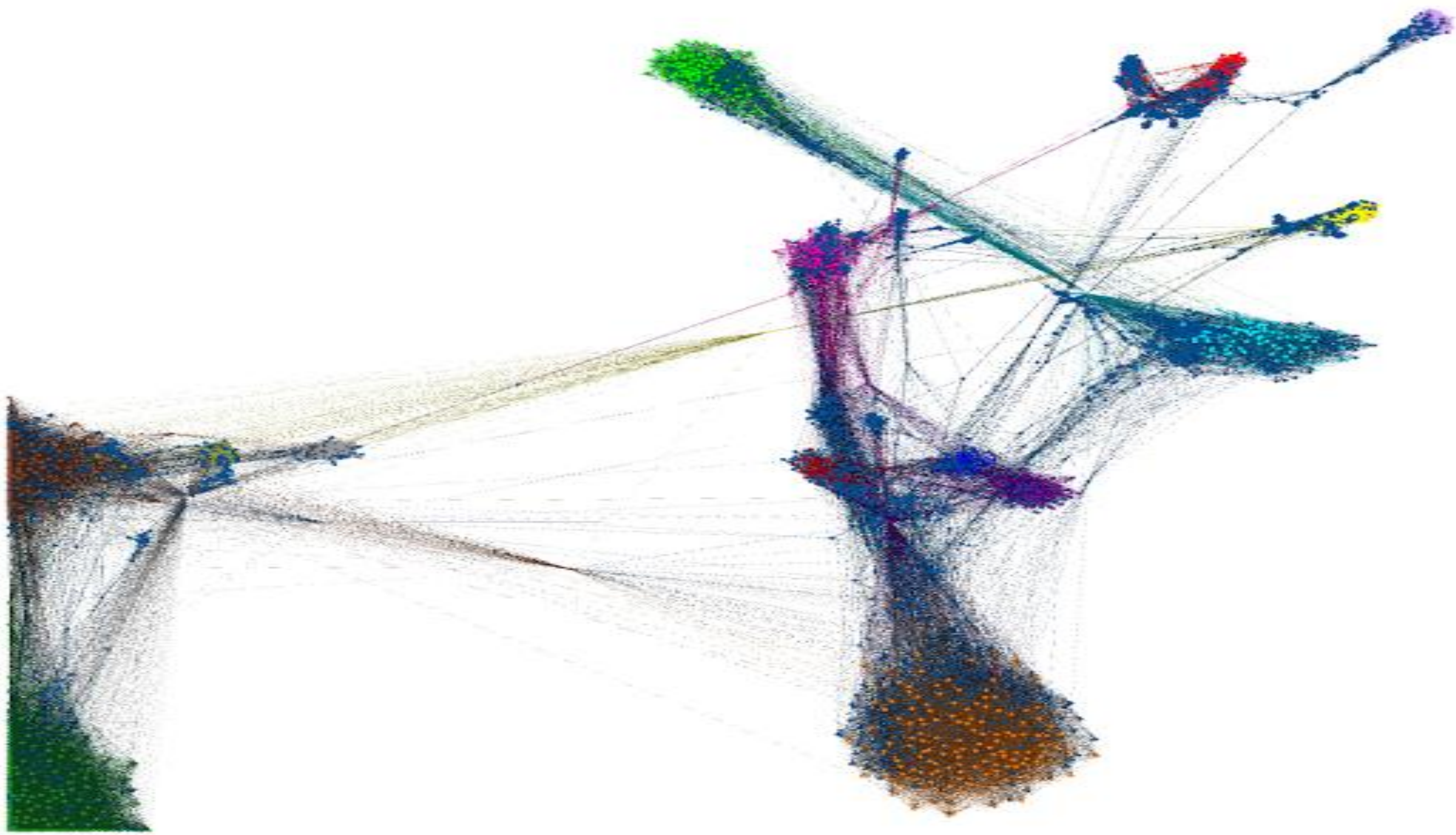
- **Idea:** Produce a k-clustering of the graph by running k-means on the effective resistance embedding

- Produced clusterings appear to have better properties than clusterings based on geometric embeddings using the k-first eigenvectors. It is also much faster for most values of k.

Visualization of unweighted social network graph



Visualization of the same graph using effective resistances as weights



Combinatorial sketching

Spectral sketching

Linear system solvers

Better combinatorial sketching

Spectral Sparsification vs Algebraic Sketching

- The sparsifier H of G has the form

$$H = B^T S^T S B$$

- Here S is a **diagonal matrix** containing the reweight factors of the edges and B is the incidence matrix for G

- Algebraic sketching : instead of solving a regression problem with a matrix B, solve instead one with SB where S is a **sparse projection matrix**
- Goal is

$$(1-\epsilon)B^T B \leq B^T S^T S B \leq (1+\epsilon)B^T B$$

Spectral Sparsification vs Algebraic Sketching

- So, spectral graph sparsification is
  a **special instance** of algebraic sketching

- Algebraic sketching takes a **very** **tall and thin** matrix
  and finds a nearly equivalent **tall and thin** matrix

- In the graph sparsification case graph combinatorics
  allow for a much **tighter control** **on size reduction**