Towards Optimal Dynamic Graph Sparsification

Sebastian Krinninger

Department of Computer Sciences
University of Salzburg

08.11.2017

A Definition

A graph $G = (V, E)$ consists of

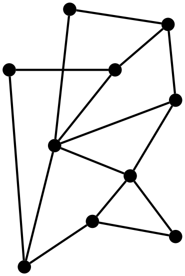
- a set of n nodes V and
- a set of m edges
  $E \subseteq \{\{u, v\} \mid u, v \in V\}$.

Graphs model **binary relationships** between entities

A Definition

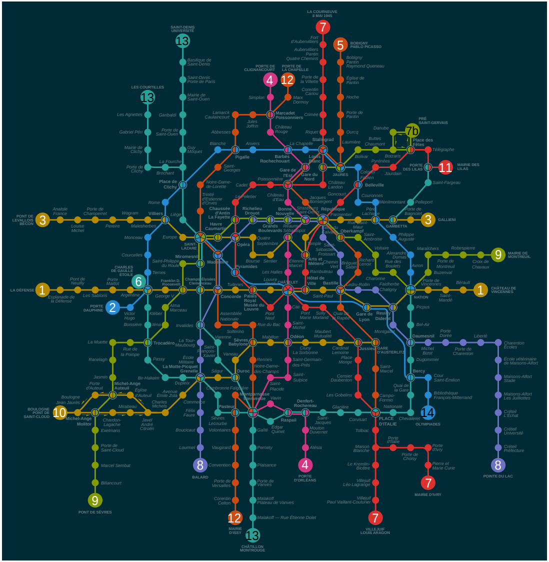
A graph $G = (V, E)$ consists of

- a set of n nodes V and
- a set of m edges
  $E \subseteq \{\{u, v\} \mid u, v \in V\}$.

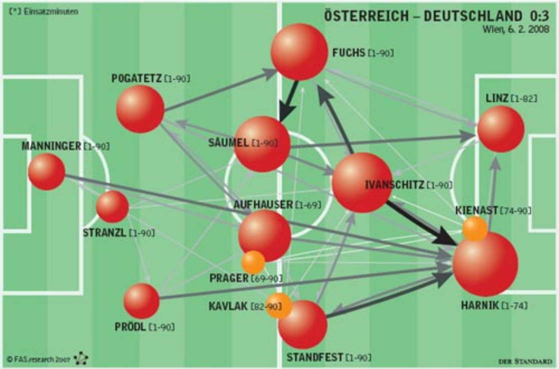


Graphs model **binary relationships** between entities

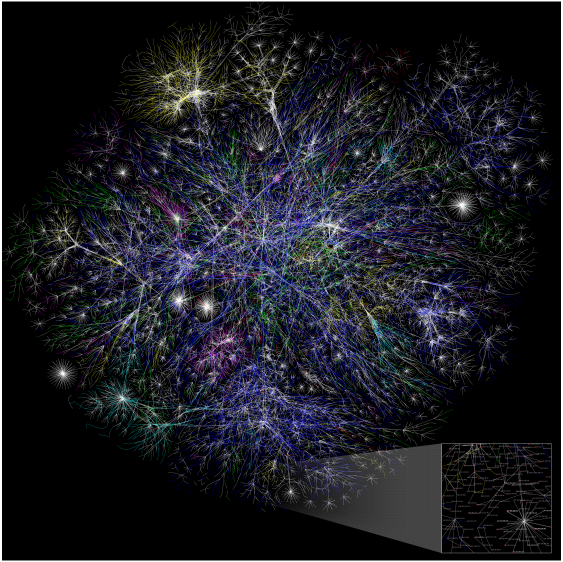
Graphs are Everywhere



Graphs are Everywhere



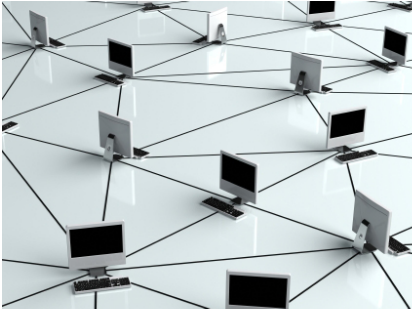
Graphs are Everywhere

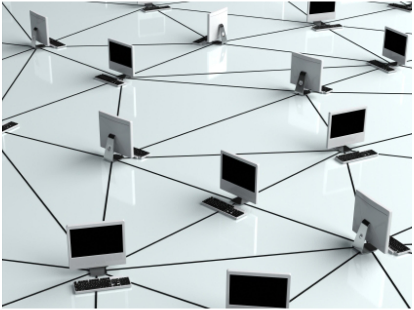


Graphs are Everywhere

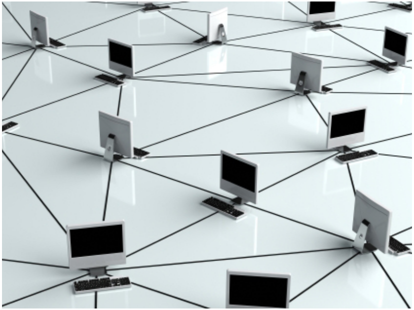
Research Area 1

Distributed and Parallel Algorithms

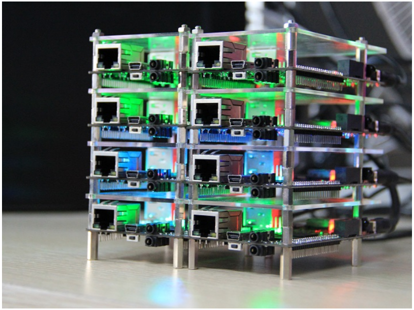


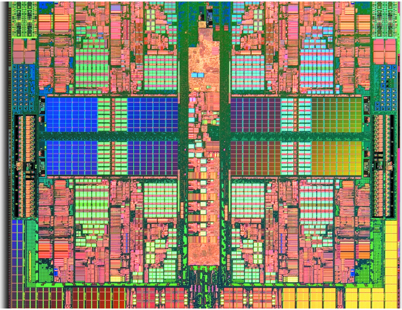



Shaw
KUKA
CITIZEN




Shaw
KUKA
CITIZEN





Shortest Path Algorithms

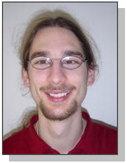
Single-Source Shortest Paths in distributed **CONGEST** model

- Improved exact algorithm
- **Close-to-optimal** approximation algorithm

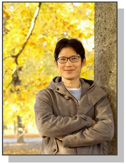
Ruben Becker

Monika Henzinger

Andreas Karrenbauer



Christoph Lenzen



Danupon Nanongkai

Shortest Path Algorithms

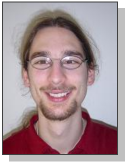
Single-Source Shortest Paths in distributed **CONGEST** model

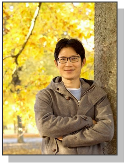
- Improved exact algorithm
- **Close-to-optimal** approximation algorithm


Ruben Becker


Monika Henzinger


Andreas Karrenbauer

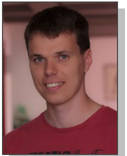

Christoph Lenzen


Danupon Nanongkai

SSSP in parallel RAM model:

- Better parallelization in presence of negative edge weights
- Improves a **sequential** problem: minimum cost-to-time ratio cycle


Karl Bringmann


Thomas Dueholm Hansen

Research Area 2

Hardness of Polynomial-Time Problems

Complexity Theory for a Big-Data World

Conventional wisdom in complexity theory (70s-90s?):

Complexity Theory for a Big-Data World

Conventional wisdom in complexity theory (70s-90s?):

- "P = tractable"
- "NP = intractable"

Complexity Theory for a Big-Data World

Conventional wisdom in complexity theory (70s-90s?):

- “P = tractable”
- “NP = intractable”
- Modulo average-case complexity, smoothed analysis, etc.

Complexity Theory for a Big-Data World

Conventional wisdom in complexity theory (70s-90s?):

- "P = tractable"
- "NP = intractable"
- Modulo average-case complexity, smoothed analysis, etc.

Reality:

- Quadratic time might be intractable
- Most desirable: (Nearly) linear time algorithms

Complexity Theory for a Big-Data World

Conventional wisdom in complexity theory (70s-90s?):

- "P = tractable"
- "NP = intractable"
- Modulo average-case complexity, smoothed analysis, etc.

Reality:

- Quadratic time might be intractable
- Most desirable: (Nearly) linear time algorithms

Prototypical Question

Can we rule out the existence of truly subquadratic time algorithms for certain problems?

Complexity Theory for a Big-Data World

Conventional wisdom in complexity theory (70s-90s?):

- "P = tractable"
- "NP = intractable"
- Modulo average-case complexity, smoothed analysis, etc.

Reality:

- Quadratic time might be intractable
- Most desirable: (Nearly) linear time algorithms

Prototypical Question

Can we rule out the existence of truly subquadratic time algorithms for certain problems?

Yes!

Complexity Theory for a Big-Data World

Conventional wisdom in complexity theory (70s-90s?):

- "P = tractable"
- "NP = intractable"
- Modulo average-case complexity, smoothed analysis, etc.

Reality:

- Quadratic time might be intractable
- Most desirable: (Nearly) linear time algorithms

Prototypical Question

Can we rule out the existence of truly subquadratic time algorithms for certain problems?

Yes! …under plausible hardness assumptions

Conditional Lower Bounds

Fine-grained complexity of **diameter approximation**

- No subquadratic algorithm under Strong Exponential Time Hypothesis [Roditty/V. Williams '13]
- Not even subquadratic $\frac{3}{2}$-approximation
- Goal: more detailed hardness analysis

Karl Bringmann

Conditional Lower Bounds

Fine-grained complexity of **diameter approximation**

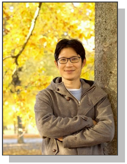
- No subquadratic algorithm under Strong Exponential Time Hypothesis [Roditty/V. Williams ’13]
- Not even subquadratic $\frac{3}{2}$-approximation
- Goal: more detailed hardness analysis

Karl Bringmann

Conditional lower bounds for **dynamic problems**

- Formulation of new hardness conjecture
- Explains certain barriers in dynamic algorithms

Monika Henzinger



Danupon Nanongkai

Thatchaphol Saranurak

Research Area 3

Dynamic Algorithms

Our World is not Static

Our World is not Static

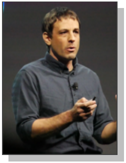
Our World is not Static

Our World is not Static

Goal: Fast recomputation of solution after update in the graph

Research on Dynamic Algorithms

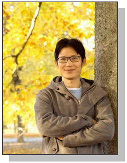
Fastest dynamic **shortest path** algorithm in a variety of settings (7+ papers)



Ittai
Abraham

Shiri
Chechik

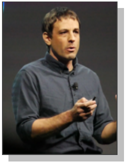
Monika
Henzinger



Danupon
Nanongkai

Research on Dynamic Algorithms

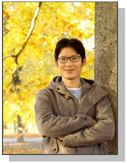
Fastest dynamic **shortest path** algorithm in a variety of settings (7+ papers)



Ittai Abraham

Shiri Chechik

Monika Henzinger

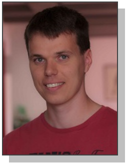


Danupon Nanongkai

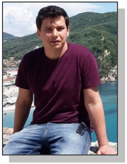
Dynamic **connectivity** and **dominators** in directed graphs

Loukas Georgiadis

Giuseppe Italiano



Thomas Dueholm Hansen



Nikos Parotsidis

Sparsification

Sparsification

Idea: Approximate dense objects by sparse objects

Sparsification

Idea: Approximate dense objects by sparse objects

Masonry arch

Sparsification

Idea: Approximate dense objects by sparse objects

Masonry arch

$\Longrightarrow$

Truss arch

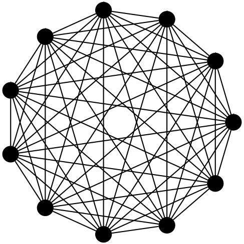
Sparsification in Graphs

Goal: Reduce to much smaller set of edges

Sparsification in Graphs

Goal: Reduce to much smaller set of edges

Dense graph

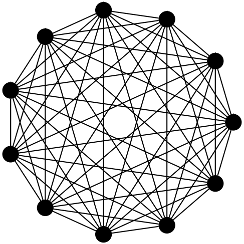


$m = \Omega(n^2)$

Sparsification in Graphs

Goal: Reduce to much smaller set of edges

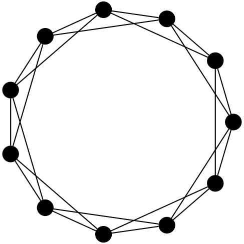
Dense graph



$m = \Omega(n^2)$

$\Longrightarrow$

Sparse graph

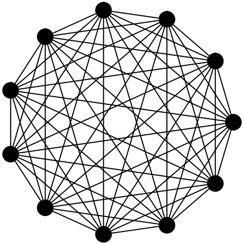


$m' \ll n^2$

Sparsification in Graphs

Goal: Reduce to much smaller set of edges

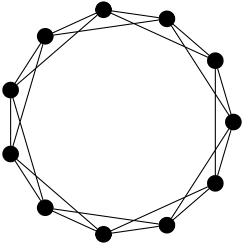
Dense graph



$m = \Omega(n^2)$

$\Longrightarrow$

Sparse graph



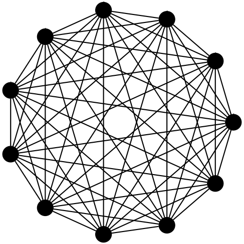
$m' \ll n^2$

Running Time: $T(n, m) \Rightarrow T(n, m')$

Sparsification in Graphs

Goal: Reduce to much smaller set of edges

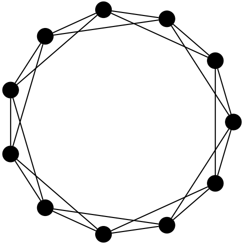
Dense graph



$m = \Omega(n^2)$

$\Longrightarrow$

Sparse graph



$m' \ll n^2$

Running Time: $T(n, m) \Rightarrow T(n, m')$

At cost of approximation

Example 1: Distance Sparsifier

Definition

A **spanner** of **stretch** α of $G = (V, E)$ is a subgraph $H = (V, E')$ such that

$$dist_G(u, v) \leq dist_H(u, v) \leq \alpha \cdot dist_G(u, v)$$

for all pairs of nodes $u, v \in V$.

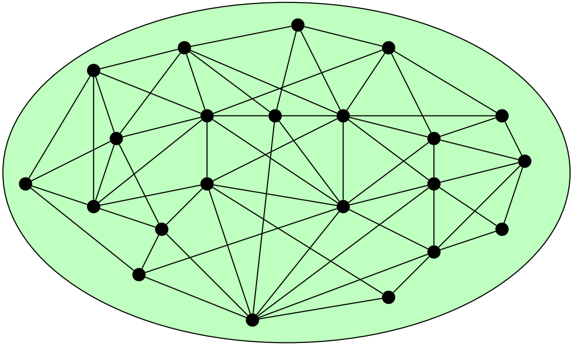
Example 1: Distance Sparsifier

Definition

A **spanner** of **stretch** α of $G = (V, E)$ is a subgraph $H = (V, E')$ such that

$$dist_G(u, v) \leq dist_H(u, v) \leq \alpha \cdot dist_G(u, v)$$

for all pairs of nodes $u, v \in V$.



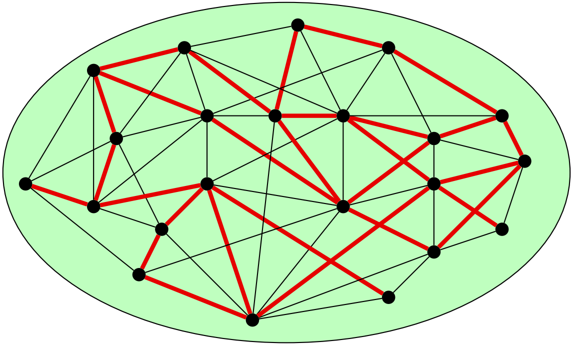
Example 1: Distance Sparsifier

Definition

A **spanner** of **stretch** α of $G = (V, E)$ is a subgraph $H = (V, E')$ such that

$$dist_G(u, v) \leq dist_H(u, v) \leq \alpha \cdot dist_G(u, v)$$

for all pairs of nodes $u, v \in V$.



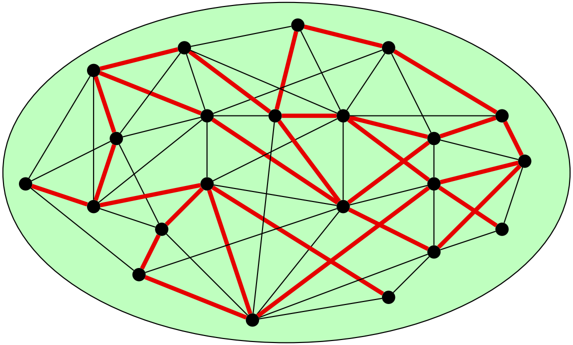
Example 1: Distance Sparsifier

Definition

A **spanner** of **stretch** α of $G = (V, E)$ is a subgraph $H = (V, E')$ such that

$$dist_G(u, v) \leq dist_H(u, v) \leq \alpha \cdot dist_G(u, v)$$

for all pairs of nodes $u, v \in V$.

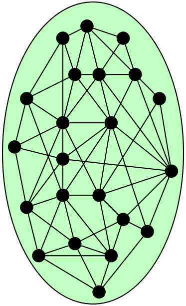


Fact: Every graph has spanners with stretch $(2k - 1)$ of size $n^{1+1/k}$ $(k \geq 2)$

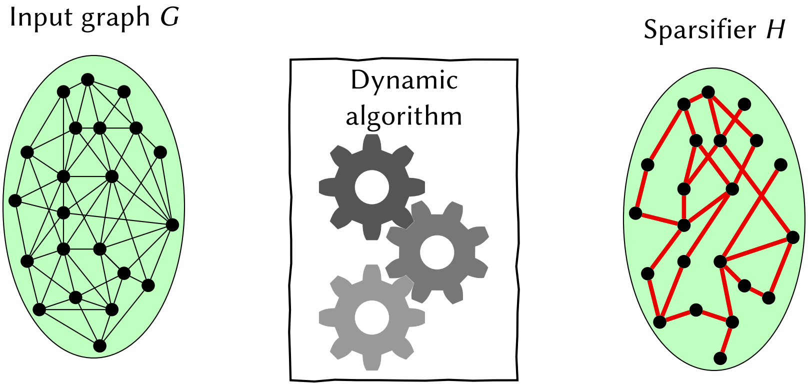
In particular: stretch $\log n$ and size $O(n)$

Dynamic Problem

Input graph G



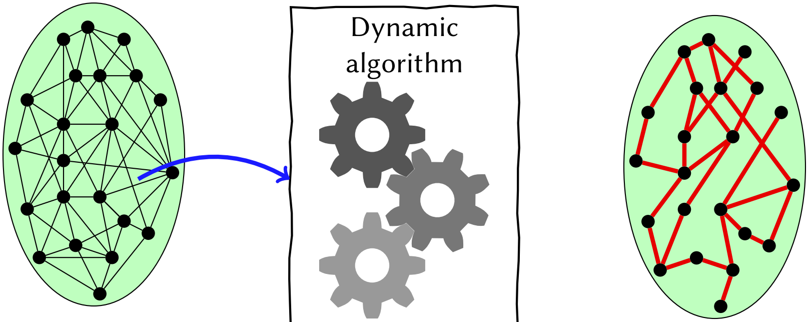
Dynamic Problem



Dynamic Problem

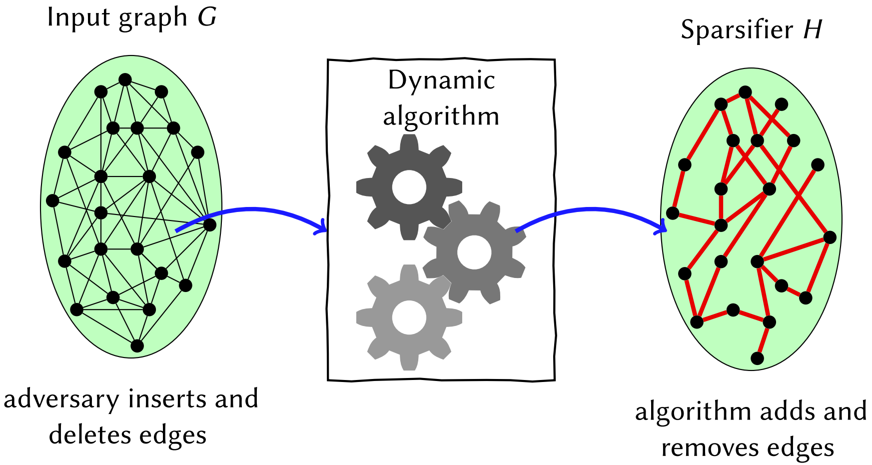
Input graph G

Sparsifier H

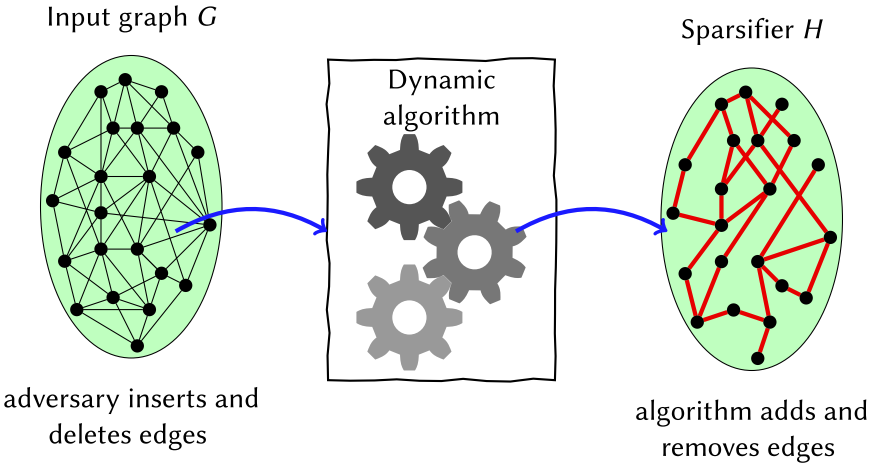


adversary inserts and deletes edges

Dynamic Problem



Dynamic Problem

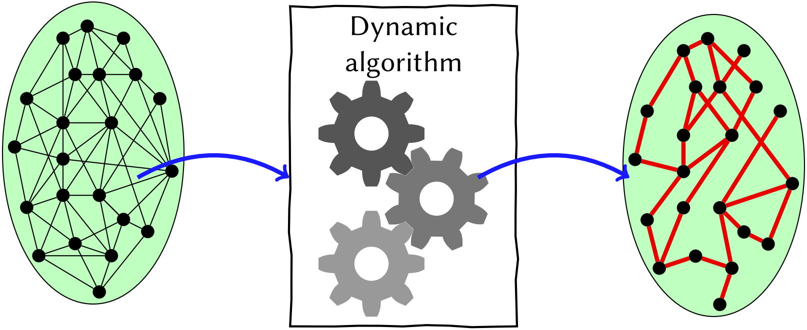


State of the art update time:

- Amortized time: $O(k^2 \log^2 n)$, stretch $2k - 1$
  Total time $O(t \cdot k^2 \log^2 n)$ for t updates [Baswana et al. 2012]

Dynamic Problem

Input graph G — Sparsifier H



adversary inserts and deletes edges

algorithm adds and removes edges

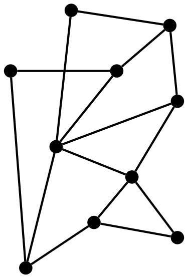
State of the art update time:

- Amortized time: $O(k^2 \log^2 n)$, stretch $2k-1$
  Total time $O(t \cdot k^2 \log^2 n)$ for t updates [Baswana et al. 2012]
- Worst-case time: $O(n^{3/4})$ for stretch 3 [Bodwin/K 2016]

Greg Bodwin

Example 2: Spectral Sparsification

View graph G as Laplacian matrix L_G

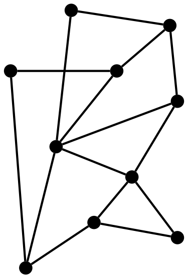


$\longrightarrow$

$$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 3 & -1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 2 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & 3 \end{pmatrix}$$

Example 2: Spectral Sparsification

View graph G as Laplacian matrix L_G



$\longrightarrow$

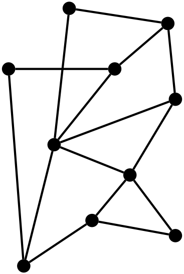
$$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 3 & -1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 2 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & 3 \end{pmatrix}$$

$$L_G[i, i] = degree(v_i)$$

$$L_G[i, j] = \begin{cases} -1 & \text{if edge } (v_i, v_j) \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$

Example 2: Spectral Sparsification

View graph G as Laplacian matrix L_G



$\rightarrow$

$$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 3 & -1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 2 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & 3 \end{pmatrix}$$

$$L_G[i,i] = degree(v_i)$$

$$L_G[i,j] = \begin{cases} -1 & \text{if edge } (v_i, v_j) \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$

Definition

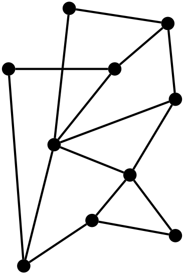
A $(1 \pm \varepsilon)$-spectral sparsifier of G is a weighted subgraph H such that

$$(1-\varepsilon)x^T L_G x \leq x^T L_H x \leq (1+\varepsilon)x^T L_G x$$

for all vectors $x \in \mathbb{R}^n$.

Example 2: Spectral Sparsification

View graph G as Laplacian matrix L_G



$\longrightarrow$

$$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 3 & -1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 2 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & 3 \end{pmatrix}$$

$$L_G[i,i] = degree(v_i)$$

$$L_G[i,j] = \begin{cases} -1 & \text{if edge } (v_i, v_j) \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$

Definition

A $(1 \pm \varepsilon)$-spectral sparsifier of G is a weighted subgraph H such that

$$(1-\varepsilon)x^T L_G x \le x^T L_H x \le (1+\varepsilon)x^T L_G x$$

for all vectors $x \in \mathbb{R}^n$.

Under Löwner ordering: $(1-\varepsilon)L_G \preceq L_H \preceq (1+\varepsilon)L_G$

Motivation I: Cut Sparsification

Consider set of nodes $S \subseteq V$ and vector $x \in \mathbb{R}^n$ such that

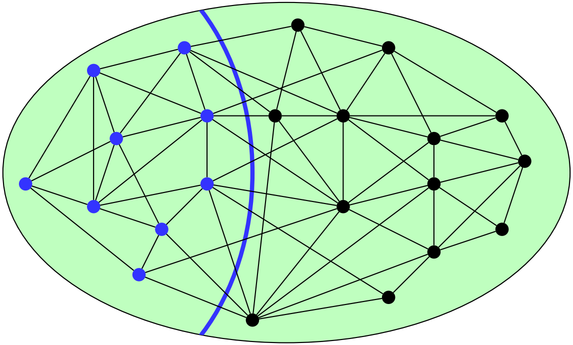
$$x_i = 1 \text{ if } i\text{-th node in } S$$
$$x_i = 0 \text{ otherwise}$$

Motivation I: Cut Sparsification

Consider set of nodes $S \subseteq V$ and vector $x \in \mathbb{R}^n$ such that

$$x_i = 1 \text{ if } i\text{-th node in } S$$
$$x_i = 0 \text{ otherwise}$$

x encodes **cut** in graph induced by S



Motivation I: Cut Sparsification

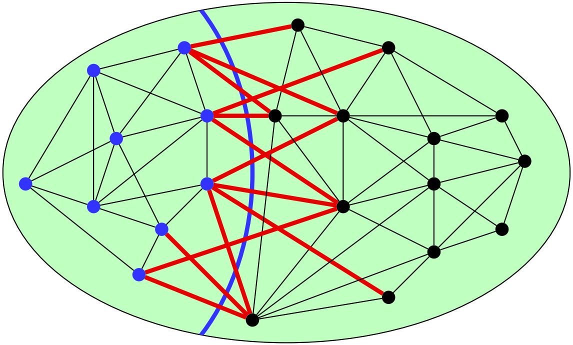
Consider set of nodes $S \subseteq V$ and vector $x \in \mathbb{R}^n$ such that

$$x_i = 1 \text{ if } i\text{-th node in } S$$

$$x_i = 0 \text{ otherwise}$$

x encodes **cut** in graph induced by S

$x^T L_G x$ corresponds to size of cut $(S, V \setminus S)$ in G



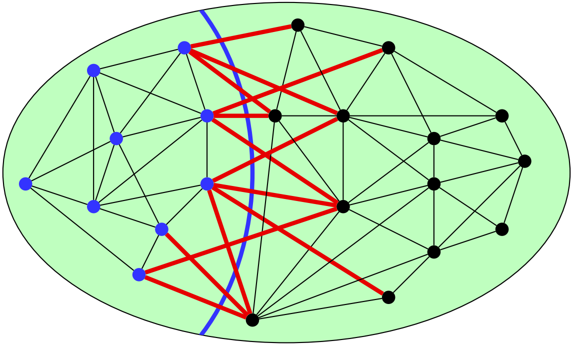
Motivation I: Cut Sparsification

Consider set of nodes $S \subseteq V$ and vector $x \in \mathbb{R}^n$ such that

$$x_i = 1 \text{ if } i\text{-th node in } S$$

$$x_i = 0 \text{ otherwise}$$

x encodes **cut** in graph induced by S
$x^T L_G x$ corresponds to size of cut $(S, V \setminus S)$ in G



⇒ Spectral sparsifier is also a cut sparsifier [Benczúr/Karger '00]

Motivation II: Solving SDD Systems

System of linear equations with *n* unknowns:

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n
\end{aligned}
$$

Motivation II: Solving SDD Systems

System of linear equations with n unknowns:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

Short: $Ax = b$, where $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$ and unknown $x \in \mathbb{R}^n$

Motivation II: Solving SDD Systems

System of linear equations with n unknowns:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

Short: $Ax = b$, where $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$ and unknown $x \in \mathbb{R}^n$

If A is **symmetric diagonally dominant (SDD)**:

- Can reduce to $L_G x = b$ with Laplacian matrix L_G of some graph G

Motivation II: Solving SDD Systems

System of linear equations with n unknowns:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

Short: $Ax = b$, where $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$ and unknown $x \in \mathbb{R}^n$

If A is **symmetric diagonally dominant (SDD)**:

- Can reduce to $L_G x = b$ with Laplacian matrix L_G of some graph G
- Amounts to computing **electrical flow** in resistor network G
  Dual formulation: $\max_{x\in\mathbb{R}^n}(2x^T b - x^T L_G x)$

Motivation II: Solving SDD Systems

System of linear equations with n unknowns:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

Short: $Ax = b$, where $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$ and unknown $x \in \mathbb{R}^n$

If A is **symmetric diagonally dominant (SDD)**:

- Can reduce to $L_G x = b$ with Laplacian matrix L_G of some graph G
- Amounts to computing **electrical flow** in resistor network G
  Dual formulation: $\max_{x \in \mathbb{R}^n} (2x^T b - x^T L_G x)$
- Nearly-linear time solvers in static setting [Spielman/Teng ’04, …]

Motivation II: Solving SDD Systems

System of linear equations with n unknowns:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

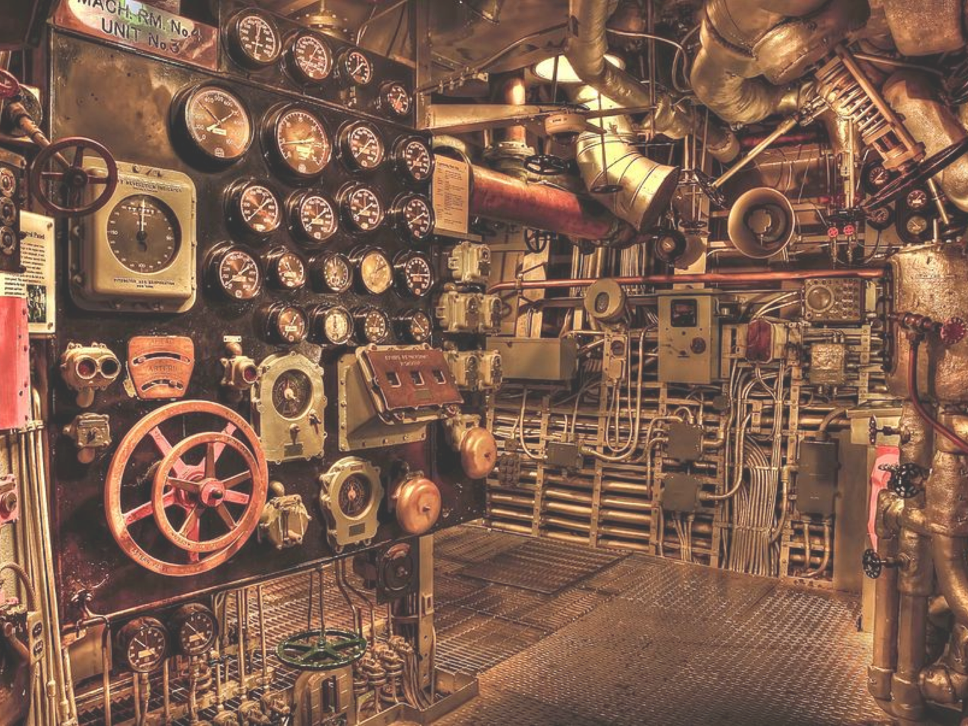
Short: $Ax = b$, where $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$ and unknown $x \in \mathbb{R}^n$

If A is **symmetric diagonally dominant (SDD)**:

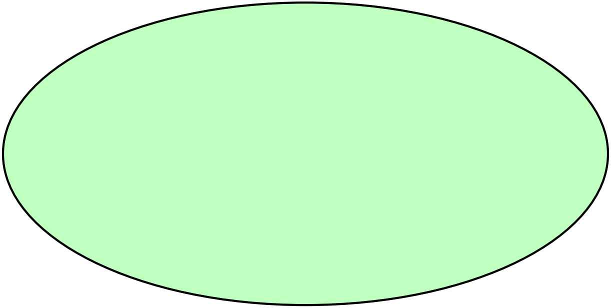
- Can reduce to $L_G x = b$ with Laplacian matrix L_G of some graph G
- Amounts to computing **electrical flow** in resistor network G
  Dual formulation: $\max_{x \in \mathbb{R}^n} (2x^T b - x^T L_G x)$
- Nearly-linear time solvers in static setting [Spielman/Teng '04, ...]

Dynamic Solver?

Changing one row in $A \rightarrow$ changing $\leq 2n$ edges of G

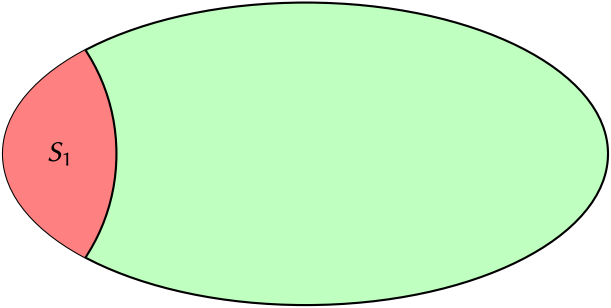

MACH. RM. No.4
UNIT No.3

t-Bundle Spanners



Idea: Pack graph with t edge-disjoint spanners of stretch $\log n$

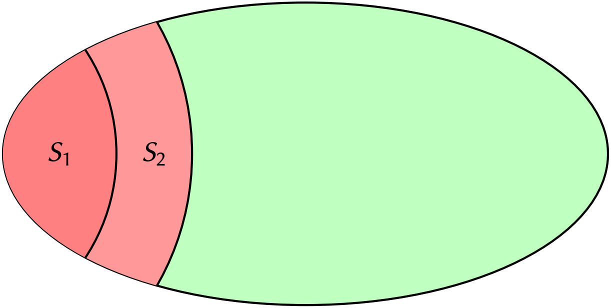
t-Bundle Spanners



Idea: Pack graph with t edge-disjoint spanners of stretch $\log n$

- Compute spanner S_1 of G

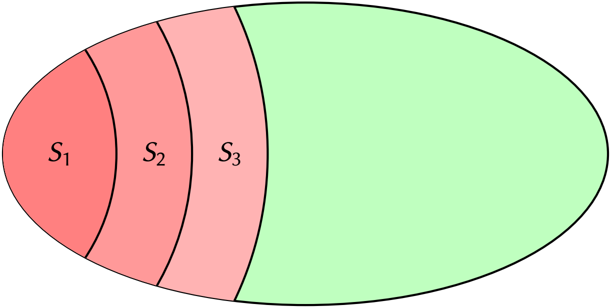
t-Bundle Spanners



Idea: Pack graph with t edge-disjoint spanners of stretch $\log n$

- Compute spanner S_1 of G
- Compute spanner S_2 of $G \setminus S_1$

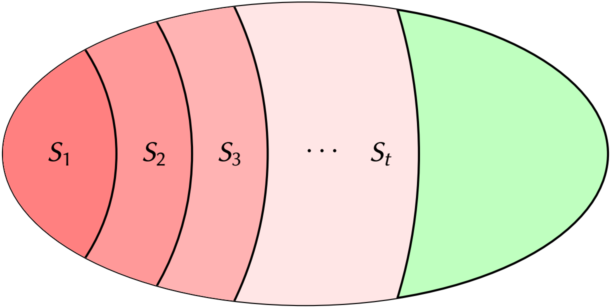
t-Bundle Spanners



Idea: Pack graph with t edge-disjoint spanners of stretch $\log n$

- Compute spanner S_1 of G
- Compute spanner S_2 of $G \setminus S_1$
- Compute spanner S_3 of $G \setminus (S_1 \cup S_2)$

t-Bundle Spanners



Idea: Pack graph with t edge-disjoint spanners of stretch $\log n$

- Compute spanner S_1 of G
- Compute spanner S_2 of $G \setminus S_1$
- Compute spanner S_3 of $G \setminus (S_1 \cup S_2)$

  $\vdots$

- Compute spanner S_t of $G \setminus (S_1 \cup S_2 \cup \cdots \cup S_{t-1})$

$B := S_1 \cup S_2 \cup \cdots \cup S_{t-1}$ is a t-**bundle spanner**

1-Step Spectral Sparsification

1. Compute t-bundle spanner B with $t = \frac{24 \log^2 n}{\varepsilon^2}$
2. Set $H = B$
3. For each edge $e \in G \setminus B$: with probability $\frac{1}{4}$, add e to H and set $w_H(e) = 4w_G(e)$

1-Step Spectral Sparsification

1. Compute t-bundle spanner B with $t = \frac{24 \log^2 n}{\varepsilon^2}$
2. Set $H = B$
3. For each edge $e \in G \setminus B$: with probability $\frac{1}{4}$, add e to H and set $w_H(e) = 4w_G(e)$

Lemma

H is a $(1 \pm \varepsilon)$-spectral sparsifier of expected size $O(n\varepsilon^{-2} \log^2 n) + m/4$.

1-Step Spectral Sparsification

1. Compute t-bundle spanner B with $t = \frac{24 \log^2 n}{\varepsilon^2}$
2. Set $H = B$
3. For each edge $e \in G \setminus B$: with probability $\frac{1}{4}$, add e to H and set $w_H(e) = 4w_G(e)$

Lemma

H is a $(1 \pm \varepsilon)$-spectral sparsifier of expected size $O(n\varepsilon^{-2} \log^2 n) + m/4$.

Intuition:

- Edges in $G \setminus B$ have small "importance" in G:

1-Step Spectral Sparsification

1. Compute t-bundle spanner B with $t = \frac{24 \log^2 n}{\varepsilon^2}$
2. Set $H = B$
3. For each edge $e \in G \setminus B$: with probability $\frac{1}{4}$, add e to H and set $w_H(e) = 4 w_G(e)$

Lemma

H is a $(1 \pm \varepsilon)$-spectral sparsifier of expected size $O(n\varepsilon^{-2} \log^2 n) + m/4$.

Intuition:

- Edges in $G \setminus B$ have small "importance" in G:
  many alternative paths of small length in B between endpoints of edge

1-Step Spectral Sparsification

1. Compute t-bundle spanner B with $t = \frac{24 \log^2 n}{\varepsilon^2}$
2. Set $H = B$
3. For each edge $e \in G \setminus B$: with probability $\frac{1}{4}$, add e to H and set $w_H(e) = 4 w_G(e)$

Lemma

H is a $(1 \pm \varepsilon)$-spectral sparsifier of expected size $O(n\varepsilon^{-2} \log^2 n) + m/4$.

Intuition:

- Edges in $G \setminus B$ have small "importance" in G:
  many alternative paths of small length in B between endpoints of edge
- Formally: B certifies small **effective resistance** of edges in $G \setminus B$

1-Step Spectral Sparsification

1. Compute t-bundle spanner B with $t = \frac{24 \log^2 n}{\varepsilon^2}$
2. Set $H = B$
3. For each edge $e \in G \setminus B$: with probability $\frac{1}{4}$, add e to H and set $w_H(e) = 4 w_G(e)$

Lemma

H is a $(1 \pm \varepsilon)$-spectral sparsifier of expected size $O(n\varepsilon^{-2} \log^2 n) + m/4$.

Intuition:

- Edges in $G \setminus B$ have small “importance” in G:
  many alternative paths of small length in B between endpoints of edge
- Formally: B certifies small **effective resistance** of edges in $G \setminus B$
- Sparsification by effective-resistance sampling [Spielman/Srivastava ’08]

1-Step Spectral Sparsification

1. Compute t-bundle spanner B with $t = \frac{24 \log^2 n}{\varepsilon^2}$
2. Set $H = B$
3. For each edge $e \in G \setminus B$: with probability $\frac{1}{4}$, add e to H and set $w_H(e) = 4 w_G(e)$

Lemma

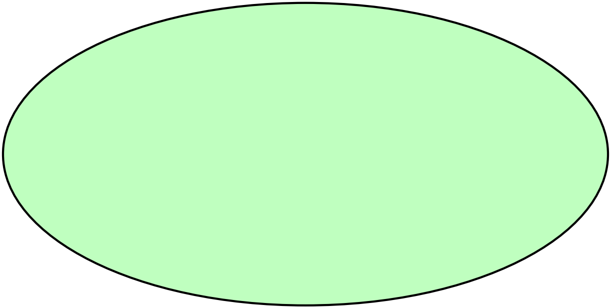
H is a $(1 \pm \varepsilon)$-spectral sparsifier of expected size $O(n\varepsilon^{-2} \log^2 n) + m/4$.

Intuition:

- Edges in $G \setminus B$ have small "importance" in G:
  many alternative paths of small length in B between endpoints of edge
- Formally: B certifies small **effective resistance** of edges in $G \setminus B$
- Sparsification by effective-resistance sampling [Spielman/Srivastava '08]
- Technical tool: concentration bounds for random matrices

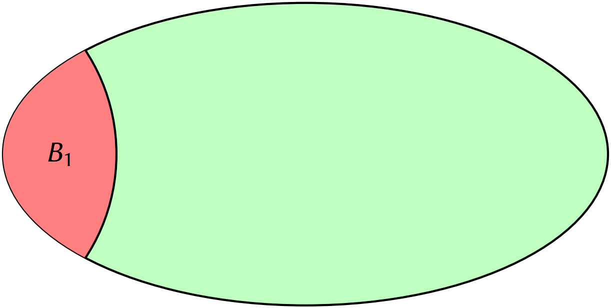
Spectral Sparsification Algorithm [Koutis '14]

Repeat 1-step sparsification on remaining graph until size is small enough



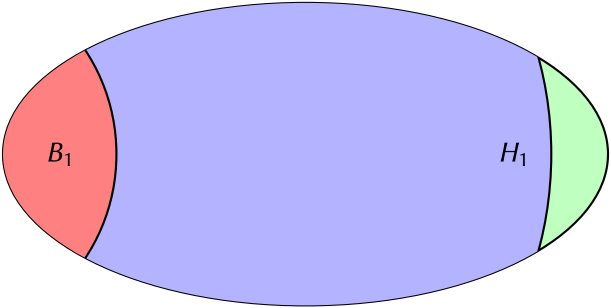
Spectral Sparsification Algorithm [Koutis '14]

Repeat 1-step sparsification on remaining graph until size is small enough



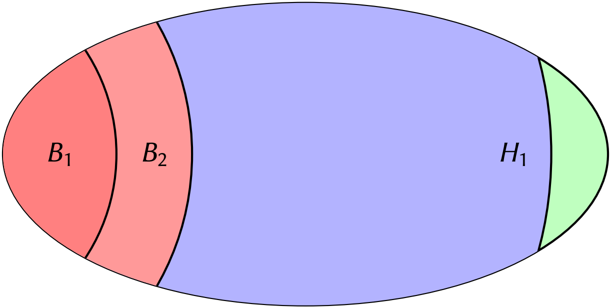
Spectral Sparsification Algorithm [Koutis '14]

Repeat 1-step sparsification on remaining graph until size is small enough



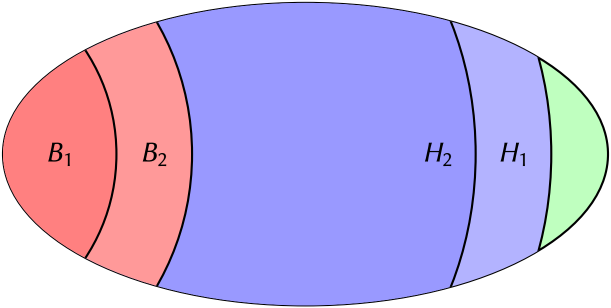
Spectral Sparsification Algorithm [Koutis '14]

Repeat 1-step sparsification on remaining graph until size is small enough



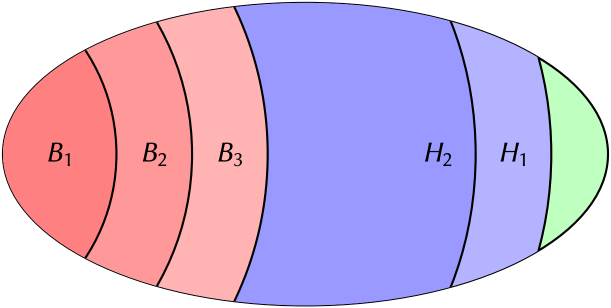
Spectral Sparsification Algorithm [Koutis '14]

Repeat 1-step sparsification on remaining graph until size is small enough



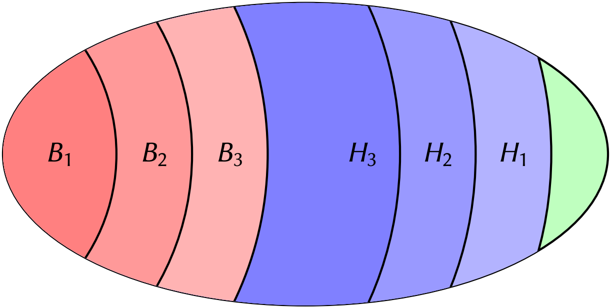
Spectral Sparsification Algorithm [Koutis '14]

Repeat 1-step sparsification on remaining graph until size is small enough



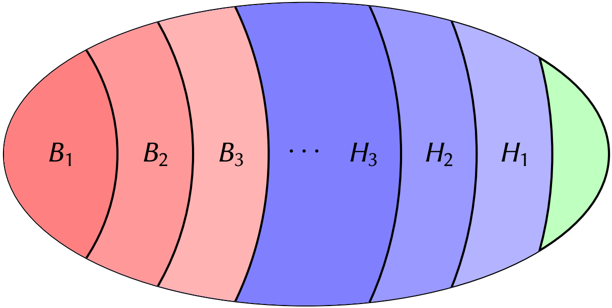
Spectral Sparsification Algorithm [Koutis '14]

Repeat 1-step sparsification on remaining graph until size is small enough



Spectral Sparsification Algorithm [Koutis '14]

Repeat 1-step sparsification on remaining graph until size is small enough

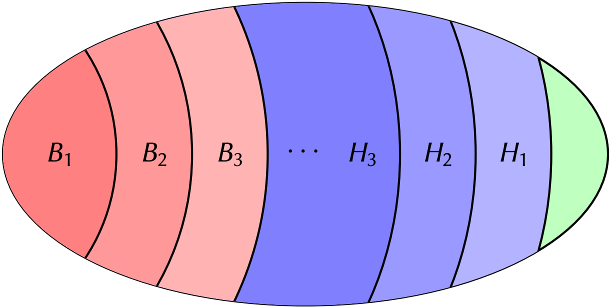


After $k = \Theta(\log n)$ iterations:

- Size of sparsifier: $O(kn\varepsilon^{-2}\log^2 n + m/4^k) = O(n\varepsilon^{-2}\log^3 n)$

Spectral Sparsification Algorithm [Koutis '14]

Repeat 1-step sparsification on remaining graph until size is small enough

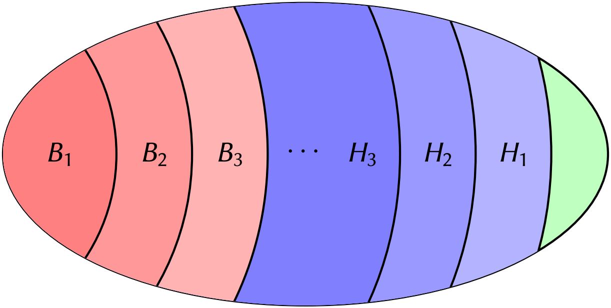


After $k = \Theta(\log n)$ iterations:

- Size of sparsifier: $O(kn\varepsilon^{-2}\log^2 n + m/4^k) = O(n\varepsilon^{-2}\log^3 n)$
- Multiplicative error: $(1 \pm \varepsilon)^{\log n}$
  Run with increased precision $\varepsilon' = \varepsilon/(2\log n)$ to ensure $(1 \pm \varepsilon)$-error

Spectral Sparsification Algorithm [Koutis '14]

Repeat 1-step sparsification on remaining graph until size is small enough



After $k = \Theta(\log n)$ iterations:

- Size of sparsifier: $O(kn\varepsilon^{-2}\log^2 n + m/4^k) = O(n\varepsilon^{-2}\log^3 n)$
- Multiplicative error: $(1 \pm \varepsilon)^{\log n}$
  Run with increased precision $\varepsilon' = \varepsilon/(2\log n)$ to ensure $(1 \pm \varepsilon)$-error

Good parallelization due to parallel spanner algorithm [Baswana/Sen '03]

Towards a Dynamic Algorithm

Straightforward approach:

- Use dynamic spanner algorithm
- Sampling is simple anyway

Towards a Dynamic Algorithm

Straightforward approach:

- Use dynamic spanner algorithm
- Sampling is simple anyway

Problem: Changes might propagate exponentially!

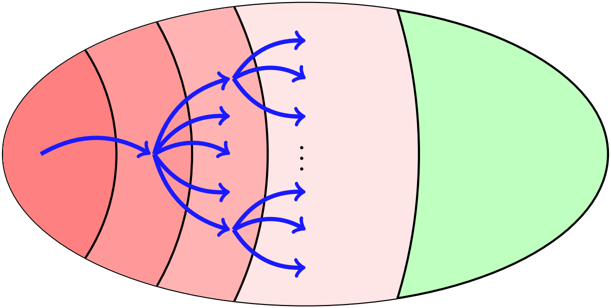
Towards a Dynamic Algorithm

Straightforward approach:

- Use dynamic spanner algorithm
- Sampling is simple anyway

Problem: Changes might propagate exponentially!

One update in G might result in $(\log n)^t$ changes to the t-bundle spanner



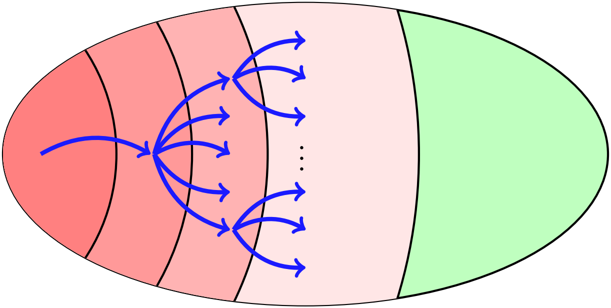
Towards a Dynamic Algorithm

Straightforward approach:

- Use dynamic spanner algorithm
- Sampling is simple anyway

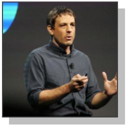
Problem: Changes might propagate exponentially!

One update in G might result in $(\log n)^t$ changes to the t-bundle spanner



Solution: Refined algorithm design

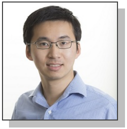
Our Algorithm



Ittai Abraham

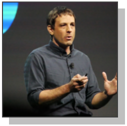
David Durfee

Ioannis Koutis



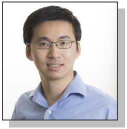
Richard Peng

Our Algorithm


Ittai Abraham


David Durfee

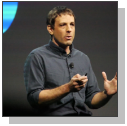

Ioannis Koutis


Richard Peng

Careful orchestration:

- Restrict to edge deletions only, amortize over sequence of deletions
  Reduction to turn deletions-only sparsifier into fully dynamic sparsifier

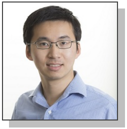
Our Algorithm



Ittai Abraham

David Durfee

Ioannis Koutis

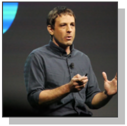


Richard Peng

Careful orchestration:

- Restrict to edge deletions only, amortize over sequence of deletions
  Reduction to turn deletions-only sparsifier into fully dynamic sparsifier
- Monotonicity property: Every edge added to the spanner S by the algorithm stays in S until deleted from input graph G
- If G only sees edge deletions, then also $G \setminus S$ only sees edge deletions

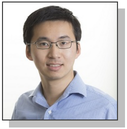
Our Algorithm



Ittai Abraham

David Durfee

Ioannis Koutis

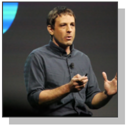


Richard Peng

Careful orchestration:

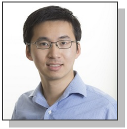
- Restrict to edge deletions only, amortize over sequence of deletions
  Reduction to turn deletions-only sparsifier into fully dynamic sparsifier
- Monotonicity property: Every edge added to the spanner S by the algorithm stays in S until deleted from input graph G
- If G only sees edge deletions, then also $G \setminus S$ only sees edge deletions
- **Challenge:** Modify Baswana et al. spanner to ensure monotonicity

Our Algorithm


Ittai Abraham


David Durfee


Ioannis Koutis


Richard Peng

Careful orchestration:

- Restrict to edge deletions only, amortize over sequence of deletions
  Reduction to turn deletions-only sparsifier into fully dynamic sparsifier
- Monotonicity property: Every edge added to the spanner S by the algorithm stays in S until deleted from input graph G
- If G only sees edge deletions, then also $G \setminus S$ only sees edge deletions
- **Challenge:** Modify Baswana et al. spanner to ensure monotonicity

Theorem (Abraham et al. '16)

There is a dynamic algorithm for maintaining a spectral sparsifier of size $n \cdot poly(\log n, \varepsilon^{-1})$ with amortized update time $poly(\log n, \varepsilon^{-1})$ per edge insertion/deletion.

Conclusion

Sparsification is a

- mathematically clean framework

Conclusion

Sparsification is a

- mathematically clean framework
- powerful tool in modern algorithm design

Conclusion

Sparsification is a

- mathematically clean framework
- powerful tool in modern algorithm design

My goal:

- Tighten connection between dynamic graph algorithms and combinatorial/continuous optimization
- Rebuild “sparsification infrastructure” in the dynamic world

Conclusion

Sparsification is a

- mathematically clean framework
- powerful tool in modern algorithm design

My goal:

- Tighten connection between dynamic graph algorithms and combinatorial/continuous optimization
- Rebuild "sparsification infrastructure" in the dynamic world

Thank you!

Conclusion

Sparsification is a

- mathematically clean framework
- powerful tool in modern algorithm design

My goal:

- Tighten connection between dynamic graph algorithms and combinatorial/continuous optimization
- Rebuild "sparsification infrastructure" in the dynamic world

Thank you!

Questions?