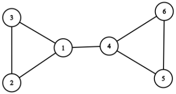
Efficient Algorithms for Graph Sparsification

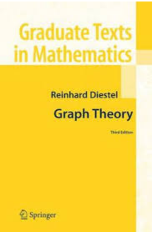
Ronald de Wolf

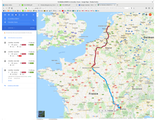
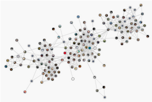
CWI QuSoft Universiteit van Amsterdam

Joint with **Simon Apers** (INRIA & CWI)

Graphs





- Big part of discrete mathematics
- Graphs model many important phenomena:
  - Logistics 
  - Internet 
  - Social networks 
- Sparse graphs are better than dense graphs:
  - Need less space to store
  - Need less time to operate on

Sparsifiers

- Graph $G = (V, E, w)$ with vertex set $V = [n]$, $m = |E|$ edges, weight function $w : E \to \mathbb{R}_{\geq 0}$. Given as adjacency list
- Laplacian of graph G: $L_G = \sum_{e \in E} w(e) L_e$

$$L_{e=(i,j)} = (e_i - e_j)(e_i - e_j)^T = \begin{pmatrix} \ddots & & \cdots & & \\ & 1 & 0 & -1 & \\ & 0 & 0 & 0 & \\ & -1 & 0 & 1 & \\ & & \cdots & & \ddots \end{pmatrix}$$

- An ε-spectral sparsifier of G is a graph $H = (V, E' \subseteq E, w')$ such that

$$\text{for all } x \in \mathbb{R}^n : x^T L_G x = (1 \pm \varepsilon) x^T L_H x$$

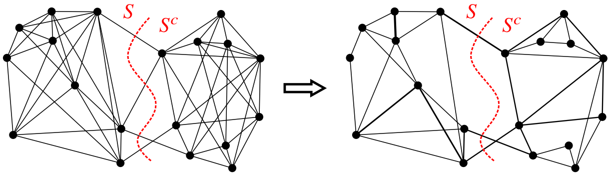
So sparsification approximately preserves all quadratic forms of the Laplacian. NB: reweighting is essential here

Sparsifiers preserve all cuts in the graph

- An ε-spectral sparsifier of $G = (V, E, w)$ is a graph $H = (V, E', w')$ s.t. for all $x \in \mathbb{R}^n$: $x^T L_G x = (1 \pm \varepsilon) x^T L_H x$
- Special case: consider $x \in \{0, 1\}^n$, with support S. $x^T L_e x = 1$ if edge e is cut, 0 otherwise. Hence

$$x^T L_G x = \sum_e w(e) x^T L_e x = \sum_{e \in S \times S^c} w(e)$$

is the value of the cut S, S^c. So H preserves all cuts of G!

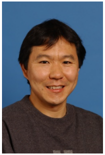


Good sparsifiers exist & are cheap to find!

An ε-spectral sparsifier of $G = (V, E, w)$ is a graph $H = (V, E', w')$ s.t. for all $x \in \mathbb{R}^n$: $x^T L_G x = (1 \pm \varepsilon) x^T L_H x$

- How sparse can we make H?
  $|E'| \approx n/\varepsilon^2$ edges are necessary and sufficient
- How quickly can we find such an H?
  Near-linear time in the input length! $\tilde{O}(m)$ time
  (input given as adjacency list for each vertex)
- Many applications

Gödel Prize 2015 for
Dan Spielman and Shang-Hua Teng



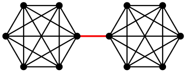
Applications (non-exhaustive list)

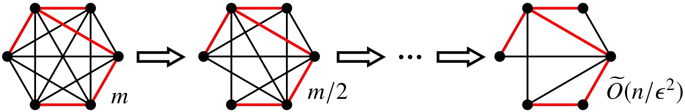
General idea to operate efficiently on graphs: first sparsify input graph G, then run your best algorithm on the sparsifier H

- Approximating min-cut in near-linear time
- Approximating max-cut up to the Goeman-Williamson ratio of 0.878 in near-linear time (better approximation is hard)
- Partitioning a graph: find min-cut and partition recursively
- **Laplacian solving**: given symmetric, diagonally-dominant $n \times n$ matrix L with m nonzero entries, and $b \in \mathbb{R}^n$, find x s.t. $Lx = b$. Can solve this approximately in time $\tilde{O}(m)$: massage L to Laplacian L_G of a graph, sparsify, solve $L_H x = b$

Remarkable: few other problems solvable in near-linear time

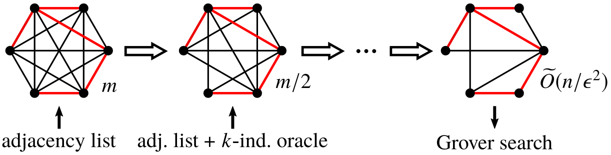
How to efficiently compute a sparsifier

- Long line of work. We'll describe approach due to Koutis-Xu'16, which repeatedly cuts number of edges in half
- Some edges are more important than others: 
- Idea: identify most important edges by finding $O(\log(n)^2/\varepsilon^2)$ disjoint spanners of G (preserve distances, linear-time comp.) Keep their edges in the sparsifier, plus a random sample of half of remaining edges reweighted by factor 2.
  This gives a sparsifier with $\approx m/2$ edges.
- Iterate this log times to reduce m to $\tilde{O}(n/\varepsilon^2)$ edges



Faster **quantum** algorithm for sparsification

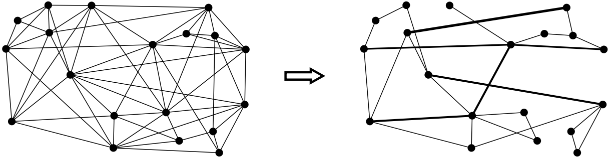
- Apers & dW'19: quantum algorithm to find ε-spectral sparsifier H in sublinear time $\tilde{O}(\sqrt{mn}/\varepsilon)$ (this is optimal!)
- Similar speed-up for cut problems, Laplacian solving etc.
- We speed up Koutis-Xu using two quantum tools: find spanners in time $O(\sqrt{mn})$, and find the final set of $\tilde{O}(n/\varepsilon^2)$ edges using Grover's quantum search algorithm



- This gives an $\tilde{O}(\sqrt{mn}/\varepsilon^2)$-algorithm. Improve ε-dependence via Spielman-Srivastava'11 (based on "effective resistances")

Summary

- Given any weighted graph G,
  in near-linear time we can compute a sparse graph H that approximately preserves most properties of G



- Leads to near-linear time algorithms for many cut problems, graph partitioning, Laplacian linear system solving, ...
- Apers & dW'19: quantum computer can do it even faster