Quantum Speedup for Graph Sparsification, Cut Approximation and Laplacian Solving

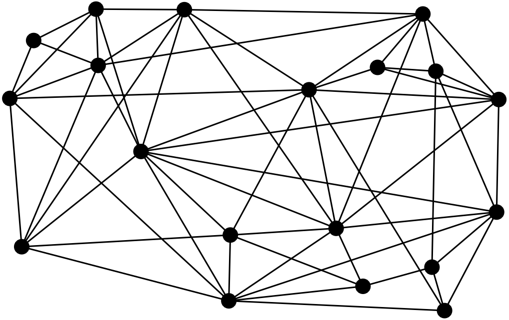
Simon Apers[1] Ronald de Wolf [2]

[1]Inria, France and CWI, the Netherlands
[2]QuSoft, CWI and University of Amsterdam, the Netherlands

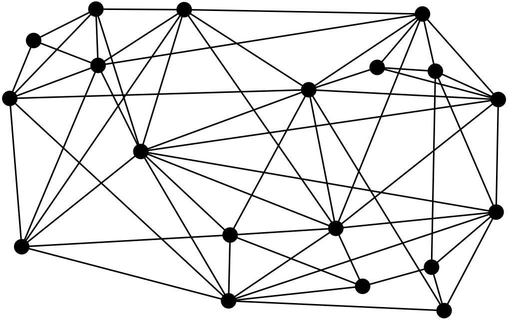
CQIF Seminar, University of Cambridge, Nov 2019

Graph Sparsification

undirected, weighted graph $G = (V, E, w)$
n nodes and $m \in O(n^2)$ edges



undirected, weighted graph $G = (V, E, w)$
n nodes and $m \in O(n^2)$ edges

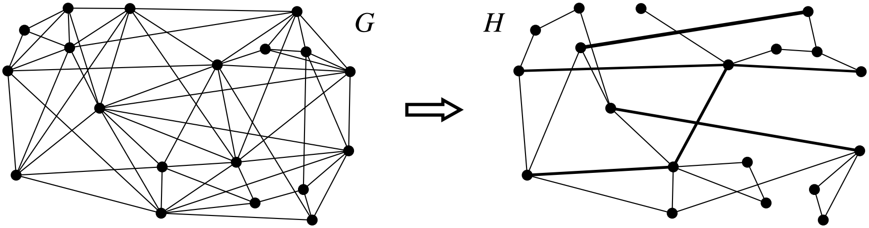


adjacency-list access: query for neighbors of nodes

Graph Sparsification

"graph sparsification"

= reduce number of edges, while preserving certain quantities



Spectral Sparsification

"graph Laplacian"

$$L_G = D - A$$

with $(D)_{ii} = \sum_j w(i,j)$, $(A)_{ij} = w(i,j)$

Spectral Sparsification

"graph Laplacian"

$$L_G = D - A$$

with $(D)_{ii} = \sum_j w(i,j)$, $(A)_{ij} = w(i,j)$

= "linear-algebraic characterization" of graph G

"quadratic forms"

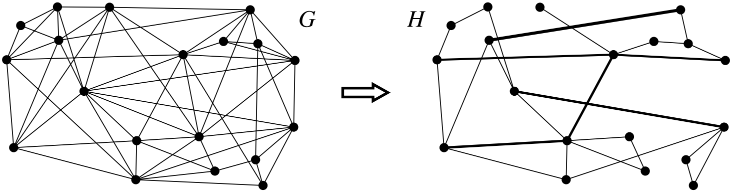
$$x^T L_G x \quad \text{and} \quad x^T L_G^+ x$$

describe cut values, eigenvalues,
effective resistances, hitting times, ...

Spectral Sparsification

"spectral sparsification"

= approximately preserve all quadratic forms

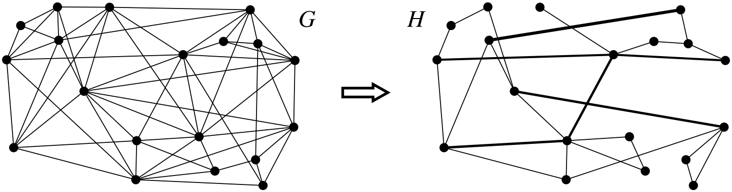


$$x^T L_H x = (1 \pm \epsilon) x_T L_G x \text{ for all } x \in \mathbb{R}^n$$

Spectral Sparsification

"spectral sparsification"

= approximately preserve all quadratic forms



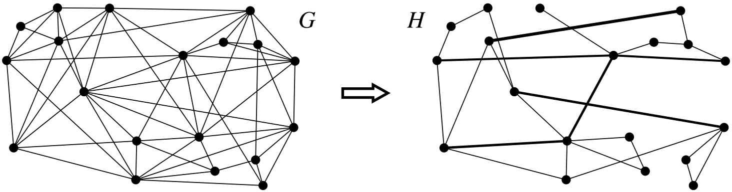
$$x^T L_H x = (1 \pm \epsilon) x_T L_G x \text{ for all } x \in \mathbb{R}^n$$

$$\Leftrightarrow x^T L_H^+ x = (1 \pm O(\epsilon)) x_T L_G^+ x$$

Spectral Sparsification

"spectral sparsification"

= approximately preserve all quadratic forms



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

$$\Leftrightarrow x^T L_H^+ x = (1 \pm O(\epsilon)) x_T L_G^+ x$$

$$\Leftrightarrow (1-\epsilon) L_G \preceq L_H \preceq (1+\epsilon) L_G$$

Spectral Sparsification

Theorem (Spielman-Teng '04)

- *every graph has ϵ-spectral sparsifier H with a number of edges*

$$\widetilde{O}(n/\epsilon^2)$$

- *H can be found in time $\widetilde{O}(m)$*

Applications

cut approximation algorithms:

$\widetilde{O}(m)$ (sparsification) + $\widetilde{O}(n)$ (approximate in H)

$\rightarrow \widetilde{O}(m)$ approximation algorithms for

- max cut (Arora-Kale'16)
- min cut (Karger'00)
- min st-cut (Peng'16)
- sparsest cut (Sherman'09)
- ...

Applications

! crucial component of Spielman-Teng breakthrough Laplacian solver:
solution to $L_H x = b$ approximates $L_G x = b$

Applications

! crucial component of Spielman-Teng breakthrough Laplacian solver: solution to $L_H x = b$ approximates $L_G x = b$

Theorem (Spielman-Teng '04)

Let G be a graph with m edges. The Laplacian system $L_G x = b$ can be solved in time $\widetilde{O}(m)$.

Applications

! crucial component of Spielman-Teng breakthrough Laplacian solver:
solution to $L_H x = b$ approximates $L_G x = b$

Theorem (Spielman-Teng '04)

Let G be a graph with m edges. The Laplacian system $L_G x = b$ can be solved in time $\widetilde{O}(m)$.

$\widetilde{O}(m)$ approximation algorithms for

- electrical flows and max flows
- spectral clustering
- random walk properties
- learning from data on graphs
- ...

Our Contribution

this work:

1. *quantum* algorithm to find ϵ-spectral sparsifier H in time

$$\widetilde{O}(\sqrt{mn}/\epsilon)$$

Our Contribution

this work:

1. *quantum* algorithm to find ϵ-spectral sparsifier H in time

$$\widetilde{O}(\sqrt{mn}/\epsilon)$$

2. matching $\widetilde{\Omega}(\sqrt{mn}/\epsilon)$ lower bound

Our Contribution

this work:

1. *quantum* algorithm to find ϵ-spectral sparsifier H in time

$$\widetilde{O}(\sqrt{mn}/\epsilon)$$

2. matching $\widetilde{\Omega}(\sqrt{mn}/\epsilon)$ lower bound
3. quantum speedup (roughly $\widetilde{O}(m)$ to $\widetilde{O}(\sqrt{mn})$) for
    - max cut, min cut, min st-cut, sparsest cut, ...
    - Laplacian solving, approximating resistances and RW properties, spectral clustering, ...

this work:

1. quantum algorithm to find H in time

$$\widetilde{O}(\sqrt{mn}/\epsilon)$$

2. matching $\widetilde{\Omega}(\sqrt{mn}/\epsilon)$ lower bound
3. quantum speedup (roughly $\widetilde{O}(m)$ to $\widetilde{O}(\sqrt{mn})$) for
   - max cut, min cut, min st-cut, sparsest cut, ...
   - Laplacian solving, approximating resistances and RW properties, spectral clustering, ...

Classical Sparsification Algorithms

Classical Sparsification Algorithms

Sparsification by edge sampling:

1. associate probabilities $\{p_e\}$ to every edge
2. keep every edge e with probability p_e, rescale its weight by $1/p_e$

Classical Sparsification Algorithms

Sparsification by edge sampling:

1. associate probabilities $\{p_e\}$ to every edge
2. keep every edge e with probability p_e, rescale its weight by $1/p_e$

ensures that

$$\mathbb{E}(w_e^H) = w_e^G, \quad \mathbb{E}(L_H) = L_G$$

Classical Sparsification Algorithms

Sparsification by edge sampling:

1. associate probabilities $\{p_e\}$ to every edge
2. keep every edge e with probability p_e, rescale its weight by $1/p_e$

ensures that

$$\mathbb{E}(w_e^H) = w_e^G, \quad \mathbb{E}(L_H) = L_G$$

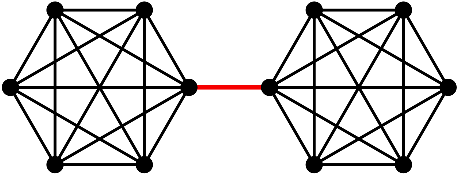
note that $\mathbb{E}(\#\text{ edges}) = \sum_e p_e \gg 1$

Classical Sparsification Algorithms

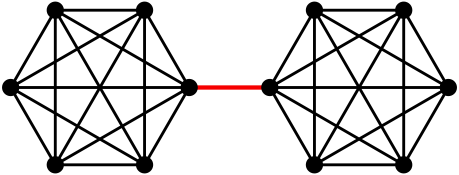
Sparsification by edge sampling:

1. associate probabilities $\{p_e\}$ to every edge
2. keep every edge e with probability p_e, rescale its weight by $1/p_e$

"important" edges should get higher p_e

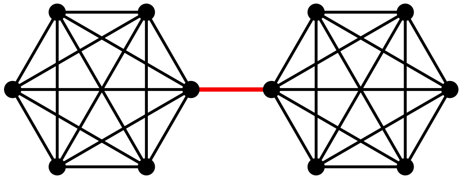


Classical Sparsification Algorithms



? how to quantify “importance” of edges ?

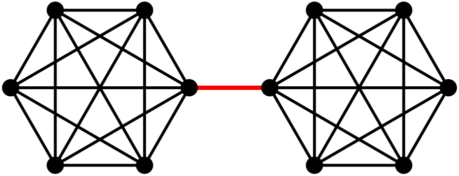
Classical Sparsification Algorithms



? how to quantify "importance" of edges ?

$\rightarrow$ high *effective resistance* R_e = effective resistance between endpoints after replacing all edges with resistors

Classical Sparsification Algorithms



? how to quantify “importance” of edges ?

$\rightarrow$ high *effective resistance* R_e = effective resistance between endpoints after replacing all edges with resistors

red edge: $R_e \in \Omega(1)$, black edges: $R_e \in O(1/n)$

Classical Sparsification Algorithms

Sparsification by edge sampling:

1. associate probabilities $\{p_e\}$ to every edge
2. keep every edge e with probability p_e, rescale its weight by $1/p_e$

Theorem (Spielman-Srivastava '08)

Setting

$$p_e = w_e R_e \log(n)/\epsilon^2$$

yields w.h.p. an ϵ-spectral sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges.

Classical Sparsification Algorithms

Sparsification by edge sampling:

1. associate probabilities $\{p_e\}$ to every edge
2. keep every edge e with probability p_e, rescale its weight by $1/p_e$

Theorem (Spielman-Srivastava '08)

Setting

$$p_e = w_e R_e \log(n)/\epsilon^2$$

yields w.h.p. an ϵ-spectral sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges.

? how to calculate R_e's ?

? how to sample according to R_e's ?

Classical Sparsification Algorithms

Alternative route:

Iterative sparsification:

1. find and keep $\widetilde{O}(n/\epsilon^2)$ "most important" edges
2. keep remaining edges independently with prob. $1/2$, and rescale weight

Classical Sparsification Algorithms

Alternative route:

Iterative sparsification:

1. find and keep $\widetilde{O}(n/\epsilon^2)$ "most important" edges
2. keep remaining edges independently with prob. $1/2$, and rescale weight

? how to find "most important" edges ?

Classical Sparsification Algorithms

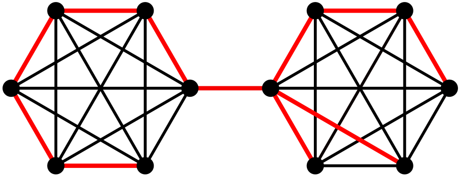
Alternative route:

Iterative sparsification:

1. find and keep $\widetilde{O}(n/\epsilon^2)$ "most important" edges
2. keep remaining edges independently with prob. $1/2$, and rescale weight

? how to find "most important" edges ?

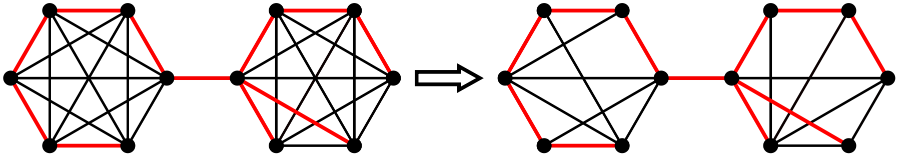
1st attempt: grow *spanning tree*
= subgraph with $n-1$ edges that connects all nodes



Classical Sparsification Algorithms

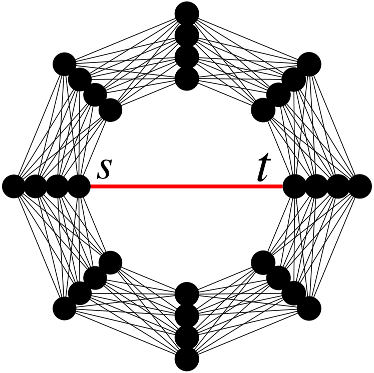
Iterative sparsification:

1. construct $\widetilde{O}(1/\epsilon^2)$ disjoint spanning trees, keep these edges
2. keep remaining edges independently with prob. $1/2$, and rescale weight



Graph Spanners

! spanning trees capture connectivity,
but not effective resistance



$$R^H(s,t) \gg R^G(s,t)$$

Graph Spanners

"graph spanner" F

Graph Spanners

"graph spanner" F

- subgraph of G with $\widetilde{O}(n)$ edges

Graph Spanners

"graph spanner" F

- subgraph of G with $\widetilde{O}(n)$ edges
- all distances stretched by factor $\log n$:

$$\mathrm{dist}_G(s,t) \le \mathrm{dist}_F(s,t) \le \log(n)\,\mathrm{dist}_G(s,t)$$

"graph spanner" F

- subgraph of G with $\widetilde{O}(n)$ edges
- all distances stretched by factor $\log n$:

$$\mathrm{dist}_G(s,t) \le \mathrm{dist}_F(s,t) \le \log(n)\,\mathrm{dist}_G(s,t)$$

$\rightarrow$ captures connectivity *and* distance

Graph Spanners

"graph spanner" F

- subgraph of G with $\widetilde{O}(n)$ edges
- all distances stretched by factor $\log n$:

$$\mathrm{dist}_G(s,t) \leq \mathrm{dist}_F(s,t) \leq \log(n)\, \mathrm{dist}_G(s,t)$$

$\rightarrow$ captures connectivity *and* distance

Theorem (Awerbuch '84)

- *Any graph G contains a spanner $F \subseteq G$ with $\widetilde{O}(n)$ edges*
- *F can be found in time $\widetilde{O}(m)$*

Classical Sparsification Algorithms

Iterative sparsification:

1. construct $\widetilde{O}(1/\epsilon^2)$ disjoint *spanners*, keep these edges
2. keep remaining edges independently with prob. $1/2$, and rescale weight

Classical Sparsification Algorithms

Iterative sparsification:

1. construct $\widetilde{O}(1/\epsilon^2)$ disjoint *spanners*, keep these edges
2. keep remaining edges independently with prob. $1/2$, and rescale weight

Theorem (Koutis-Xu '14)

W.h.p., the resulting graph is an ϵ-spectral sparsifier with a number of edges

$$m/2 + \widetilde{O}(n/\epsilon^2)$$

Classical Sparsification Algorithms

Iterative sparsification:

1. construct $\widetilde{O}(1/\epsilon^2)$ disjoint *spanners*, keep these edges
2. keep remaining edges independently with prob. $1/2$, and rescale weight

Theorem (Koutis-Xu '14)

W.h.p., the resulting graph is an ϵ-spectral sparsifier with a number of edges

$$m/2 + \widetilde{O}(n/\epsilon^2)$$

$\rightarrow$ *repeat $O(\log n)$ times:* ϵ-cut sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges

Classical Sparsification Algorithms

Iterative sparsification:

1. construct $\widetilde{O}(1/\epsilon^2)$ disjoint *spanners*, keep these edges
2. keep remaining edges independently with prob. $1/2$, and rescale weight

Theorem (Koutis-Xu '14)

W.h.p., the resulting graph is an ϵ-spectral sparsifier with a number of edges

$$m/2 + \widetilde{O}(n/\epsilon^2)$$

$\rightarrow$ *repeat $O(\log n)$ times:* ϵ-cut sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges

$\rightarrow$ spectral sparsification in time $\widetilde{O}(m/\epsilon^2)$ (can be improved to $\widetilde{O}(m)$)

Quantum Sparsification Algorithm

Quantum Sparsification Algorithm

= quantum spanner construction + k-independent advice

Quantum Sparsification Algorithm

Iterative sparsification:

1. construct $\widetilde{O}(1/\epsilon^2)$ disjoint *spanners*, keep these edges
2. keep remaining edges independently with prob. $1/2$, and rescale weight

$\rightarrow \widetilde{O}(m)$ spanner construction is main cost

Quantum Sparsification Algorithm

Iterative sparsification:

1. construct $\widetilde{O}(1/\epsilon^2)$ disjoint *spanners*, keep these edges
2. keep remaining edges independently with prob. $1/2$, and rescale weight

$\rightarrow \widetilde{O}(m)$ spanner construction is main cost

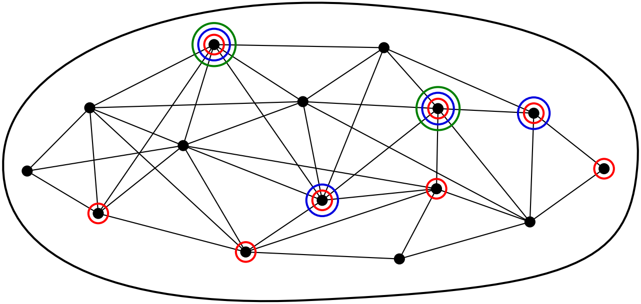
Theorem (this work)

There is a quantum algorithm that constructs a spanner in time

$$\widetilde{O}(\sqrt{mn})$$

Thorup-Zwick Spanner Algorithm

construct “centers”:

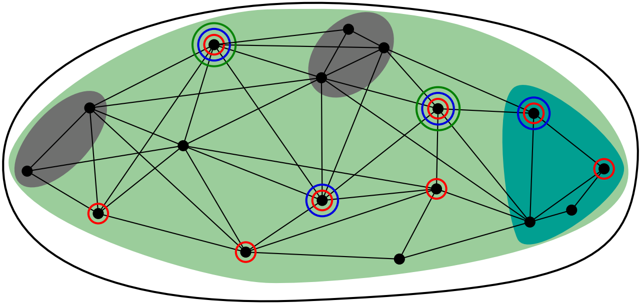


$$V = A_0 \supset \color{red}{A_1} \supset \color{blue}{A_2} \supset \cdots \supset \color{green}{A_{\log(n)-1}} \supset A_{\log(n)} = \emptyset$$

such that $A_{i+1} \subset A_i$ is random subset of size $|A_i|/2$

Thorup-Zwick Spanner Algorithm

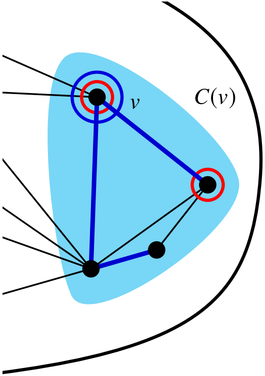
for all nodes v, route local "clusters" $C(v)$:



$$\forall v \in A_i - A_{i+1} : \quad C(v) = \{w \mid \delta(w, v) < \delta(w, A_{i+1})\}$$

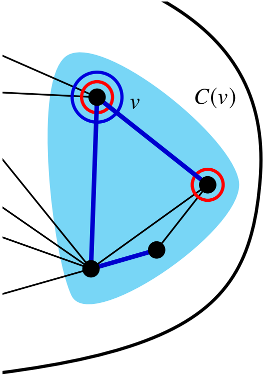
Thorup-Zwick Spanner Algorithm

for all nodes v, route local “clusters” $C(v)$:



Thorup-Zwick Spanner Algorithm

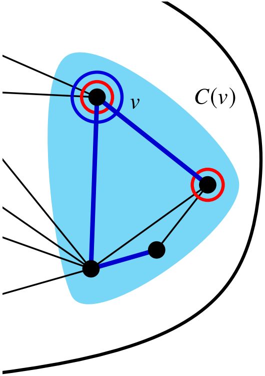
for all nodes v, route local "clusters" $C(v)$:



grow "shortest-path tree" $T(v)$ from v, spanning $C(v)$

Thorup-Zwick Spanner Algorithm

for all nodes v, route local "clusters" $C(v)$:



grow "shortest-path tree" $T(v)$ from v, spanning $C(v)$

$\rightarrow$ complexity $\widetilde{O}(|E(C(v))|)$

Thorup-Zwick Spanner Algorithm

let H be the union of all shortest-path trees:

$$H = \bigcup_v T(v)$$

Theorem (Thorup-Zwick '01)

With high probability, H is a spanner and can be constructed in time

$$\widetilde{O}\Big(\mathbb{E}\sum_v |E(C(v))|\Big) \in \widetilde{O}(m)$$

Quantum Spanner Algorithm

Theorem (Dürr-Heiligman-Høyer-Mhalla '04)

There is a quantum algorithm to construct a shortest-path tree in a graph with n nodes and m edges in time

$$\widetilde{O}(\sqrt{mn})$$

Quantum Spanner Algorithm

Theorem (Dürr-Heiligman-Høyer-Mhalla '04)

There is a quantum algorithm to construct a shortest-path tree in a graph with n nodes and m edges in time

$$\widetilde{O}(\sqrt{mn})$$

for a cluster $C(v)$ with $|C(v)|$ nodes and $|E(C(v))|$ edges:

$$\widetilde{O}(|E(C(v))|) \text{ (classical)} \longrightarrow \widetilde{O}(\sqrt{|C(v)||E(C(v))|}) \text{ (quantum)}$$

Quantum Spanner Algorithm

Theorem (Dürr-Heiligman-Høyer-Mhalla '04)

There is a quantum algorithm to construct a shortest-path tree in a graph with n nodes and m edges in time

$$\widetilde{O}(\sqrt{mn})$$

for a cluster $C(v)$ with $|C(v)|$ nodes and $|E(C(v))|$ edges:

$$\widetilde{O}(|E(C(v))|) \text{ (classical)} \longrightarrow \widetilde{O}(\sqrt{|C(v)||E(C(v))|}) \text{ (quantum)}$$

total time:

$$\widetilde{O}\Big(\sum_{v} \sqrt{|C(v)||E(C(v))|}\Big)$$

Quantum Spanner Algorithm

Thorup-Zwick spanner algorithm + DHHM SPT quantum algorithm:

total time complexity $\widetilde{O}\left(\sum_{v} \sqrt{|C(v)||E(C(v))|}\right)$

Quantum Spanner Algorithm

Thorup-Zwick spanner algorithm + DHHM SPT quantum algorithm:

total time complexity $\widetilde{O}\left(\sum_v \sqrt{|C(v)||E(C(v))|}\right)$

- Cauchy-Schwarz:

$$\widetilde{O}\left(\sum_v \sqrt{|C(v)||E(C(v))|}\right) \in \widetilde{O}\left(\sqrt{\sum_v |C(v)|}\sqrt{\sum_v |E(C(v))|}\right)$$

Quantum Spanner Algorithm

Thorup-Zwick spanner algorithm + DHHM SPT quantum algorithm:

total time complexity $\widetilde{O}\left(\sum_v \sqrt{|C(v)||E(C(v))|}\right)$

- Cauchy-Schwarz:

$$\widetilde{O}\left(\sum_v \sqrt{|C(v)||E(C(v))|}\right) \in \widetilde{O}\left(\sqrt{\sum_v |C(v)|}\sqrt{\sum_v |E(C(v))|}\right)$$

- Thorup-Zwick:

$$\mathbb{E}\left(\sum_v |C(v)|\right) \in \widetilde{O}(n), \quad \mathbb{E}\left(\sum_v |E(C(v))|\right) \in \widetilde{O}(m)$$

Quantum Spanner Algorithm

Thorup-Zwick spanner algorithm + DHHM SPT quantum algorithm:

total time complexity $\widetilde{O}\left(\sum_v \sqrt{|C(v)||E(C(v))|}\right)$

- Cauchy-Schwarz:

$$\widetilde{O}\left(\sum_v \sqrt{|C(v)||E(C(v))|}\right) \in \widetilde{O}\left(\sqrt{\sum_v |C(v)|}\sqrt{\sum_v |E(C(v))|}\right)$$

- Thorup-Zwick:

$$\mathbb{E}\left(\sum_v |C(v)|\right) \in \widetilde{O}(n), \quad \mathbb{E}\left(\sum_v |E(C(v))|\right) \in \widetilde{O}(m)$$

$\longrightarrow$ with high probability: $\widetilde{O}\left(\sum_v \sqrt{|C(v)||E(C(v))|}\right) \in \widetilde{O}(\sqrt{mn})$

Quantum Spanner Algorithm

Thorup-Zwick spanner algorithm + DHHM SPT quantum algorithm:

total time complexity $\widetilde{O}\left(\sum_v \sqrt{|C(v)||E(C(v))|}\right)$

Theorem (this work)

There is a quantum algorithm that constructs with high probability a spanner in time

$$\widetilde{O}(\sqrt{mn})$$

- Thorup-Zwick:

$$\mathbb{E}\left(\sum_v |C(v)|\right) \in \widetilde{O}(n), \quad \mathbb{E}\left(\sum_v |E(C(v))|\right) \in \widetilde{O}(m)$$

$\longrightarrow$ with high probability: $\widetilde{O}\left(\sum_v \sqrt{|C(v)||E(C(v))|}\right) \in \widetilde{O}(\sqrt{mn})$

Quantum Sparsification Algorithm

Iterative sparsification:

1. use *quantum algorithm* to construct $\widetilde{O}(1/\epsilon^2)$ disjoint *spanners*, keep these edges
2. keep remaining edges independently with prob. $1/2$, and rescale weight

Quantum Sparsification Algorithm

Iterative sparsification:

1. use *quantum algorithm* to construct $\widetilde{O}(1/\epsilon^2)$ disjoint *spanners*, keep these edges
2. keep remaining edges independently with prob. $1/2$, and rescale weight

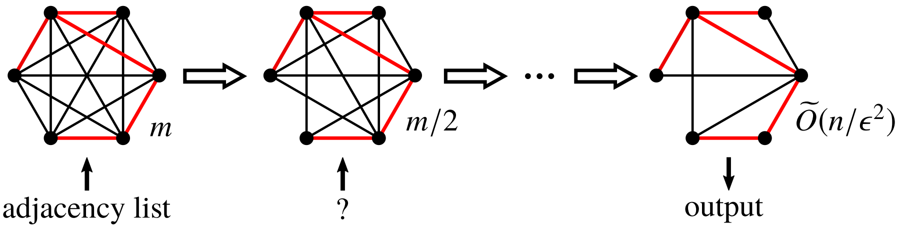
? how to keep track of “intermediate” graph with $O(m)$ edges in time $o(m)$?

Quantum Sparsification Algorithm

Iterative sparsification:

1. use *quantum algorithm* to construct $\widetilde{O}(1/\epsilon^2)$ disjoint *spanners*, keep these edges
2. keep remaining edges independently with prob. $1/2$, and rescale weight

? how to keep track of "intermediate" graph with $O(m)$ edges in time $o(m)$?



Random Advice

adjacency list query: $|i, k, 0\rangle \xrightarrow{O_G} |i, k, l\rangle$ (l is k-th neighbor of i)

Random Advice

adjacency list query: $|i, k, 0\rangle \xrightarrow{O_G} |i, k, l\rangle$ (l is k-th neighbor of i)

- add random advice string $x \in \{0,1\}^m$

1	0	0	1	1	0	1	1	1	0	1	0	0

edge i discarded edge j kept

Random Advice

adjacency list query: $|i,k,0\rangle \xrightarrow{O_G} |i,k,l\rangle$ (l is k-th neighbor of i)

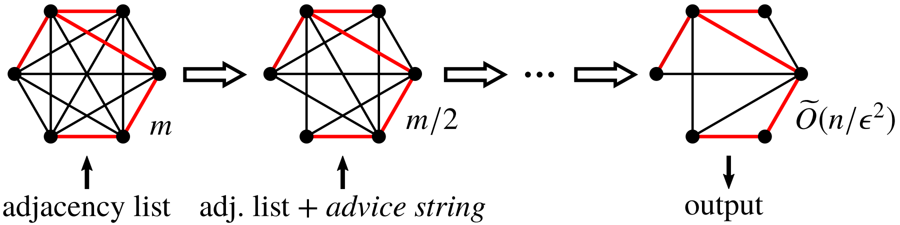
- add random advice string $x \in \{0,1\}^m$

1	0	0	1	1	0	1	1	1	0	1	0	0

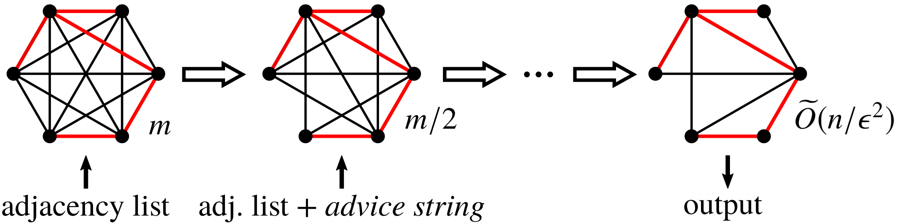
edge i discarded edge j kept

adjacency list + advice string: $|i,k,0,0\rangle \xrightarrow{O_G \otimes O_x} |i,k,l,x(k,l)\rangle$

Random Advice



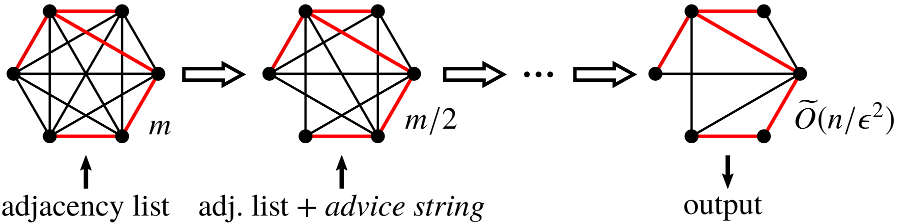
Random Advice



when encountering edge (k, l) with $x(k, l) = 0$,
set its weight/conductance to zero:

$$|k, l, 0\rangle \xrightarrow{O_G} |k, l, x(k, l) \cdot w(k, l)\rangle$$

Random Advice



when encountering edge (k,l) with $x(k,l)=0$,
set its weight/conductance to zero:

$$|k,l,0\rangle \xrightarrow{O_G} |k,l,x(k,l)\cdot w(k,l)\rangle$$

$\rightarrow$ grow spanners in intermediate graphs using adj. list + advice string

Quantum Sparsification Algorithm

Iterative sparsification:

1. use *quantum algorithm* to construct $\widetilde{O}(1/\epsilon^2)$ disjoint *spanners*, keep these edges
2. use *random advice string* to keep remaining edges independently with prob. $1/2$, and rescale weight

Quantum Sparsification Algorithm

Iterative sparsification:

1. use *quantum algorithm* to construct $\widetilde{O}(1/\epsilon^2)$ disjoint *spanners*, keep these edges
2. use *random advice string* to keep remaining edges independently with prob. $1/2$, and rescale weight

! time $O(m)$ to generate random advice string $x \in \{0,1\}^m$!

Quantum Sparsification Algorithm

Iterative sparsification:

1. use *quantum algorithm* to construct $\widetilde{O}(1/\epsilon^2)$ disjoint *spanners*, keep these edges
2. use *random advice string* to keep remaining edges independently with prob. $1/2$, and rescale weight

! time $O(m)$ to generate random advice string $x \in \{0,1\}^m$!

- classical solution: “lazy sampling” (generate bits on demand)

Quantum Sparsification Algorithm

Iterative sparsification:

1. use *quantum algorithm* to construct $\widetilde{O}(1/\epsilon^2)$ disjoint *spanners*, keep these edges
2. use *random advice string* to keep remaining edges independently with prob. $1/2$, and rescale weight

! time $O(m)$ to generate random advice string $x \in \{0,1\}^m$!

- classical solution: "lazy sampling" (generate bits on demand)
- quantum this is not possible: can address all bits in the string in superposition

Rid of Random Advice: Stage 1

k-wise independent string $x \in \{0, 1\}^m$

Rid of Random Advice: Stage 1

k-wise independent string $x \in \{0,1\}^m$

= behaves uniformly random on every subset of k bits

$$\forall S \subset [m],\ |S| \le k,\ z \in \{0,1\}^{|S|} : \quad \Pr(x_S = z) = 1/2^{|S|}$$

Rid of Random Advice: Stage 1

Fact

Quantum algorithm cannot distinguish uniformly random string from $2q$-wise independent string using $\leq q$ queries.

Rid of Random Advice: Stage 1

Fact

Quantum algorithm cannot distinguish uniformly random string from $2q$-wise independent string using $\leq q$ queries.

Proof.

Polynomial method (Beals-Buhrman-Cleve-Mosca-de Wolf '98):

> *outcome of any binary measurement after q queries can be described by multilinear polynomial in $x(i)'s$ of degree $\leq 2q$*

Rid of Random Advice: Stage 1

Fact

Quantum algorithm cannot distinguish uniformly random string from $2q$-wise independent string using $\leq q$ queries.

Proof.

Polynomial method (Beals-Buhrman-Cleve-Mosca-de Wolf '98):

> *outcome of any binary measurement after q queries can be described by multilinear polynomial in $x(i)'s$ of degree $\leq 2q$*

Expectation of every term, and thus of entire polynomial, hence depends on $\leq 2q$ input bits. This cannot distinguish $2q$-wise independent x from uniformly random x. $\square$

Rid of Random Advice: Stage 2

T-time quantum algorithm only requires $2T$-wise independent advice

Rid of Random Advice: Stage 2

T-time quantum algorithm only requires $2T$-wise independent advice

$\rightarrow$ can we efficiently simulate such advice for $T \in o(m)$?

Rid of Random Advice: Stage 2

T-time quantum algorithm only requires $2T$-wise independent advice

$\rightarrow$ can we efficiently simulate such advice for $T \in o(m)$?

classic construction: random degree-$2T$ polynomials

↯ $\Omega(T)$ time per query

Rid of Random Advice: Stage 2

luckily, there is other interest:
“efficient k-independent hash functions”

Rid of Random Advice: Stage 2

luckily, there is other interest:
"efficient k-independent hash functions"

based on new constructions of expanders and hash tables:

Theorem (Christiani-Pagh-Thorup '15)

Can construct in preprocessing time $\widetilde{O}(k)$ a k-independent oracle that simulates queries to a k-wise independent string in time $\widetilde{O}(1)$ per query.

Rid of Random Advice: Stage 2

luckily, there is other interest:
"efficient k-independent hash functions"

based on new constructions of expanders and hash tables:

Theorem (Christiani-Pagh-Thorup '15)

Can construct in preprocessing time $\widetilde{O}(k)$ a k-independent oracle that simulates queries to a k-wise independent string in time $\widetilde{O}(1)$ per query.

Corollary

Any time-T quantum algorithm using uniformly random advice can be simulated in time $\widetilde{O}(T)$ without advice.

Quantum Sparsification Algorithm

Quantum iterative sparsification:

1. use *quantum algorithm* to construct $\widetilde{O}(1/\epsilon^2)$ disjoint *spanners*, keep these edges
2. construct k-*independent oracle*
3. use oracle to “mark” remaining edges independently with prob. $1/2$, and rescale weights

Quantum Sparsification Algorithm

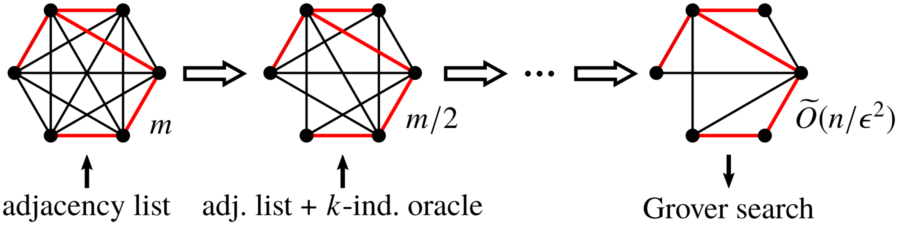
Quantum iterative sparsification:

1. use *quantum algorithm* to construct $\widetilde{O}(1/\epsilon^2)$ disjoint *spanners*, keep these edges
2. construct k*-independent oracle*
3. use oracle to “mark” remaining edges independently with prob. $1/2$, and rescale weights

$\rightarrow$ per iteration: complexity $\widetilde{O}(\sqrt{mn}/\epsilon^2)$

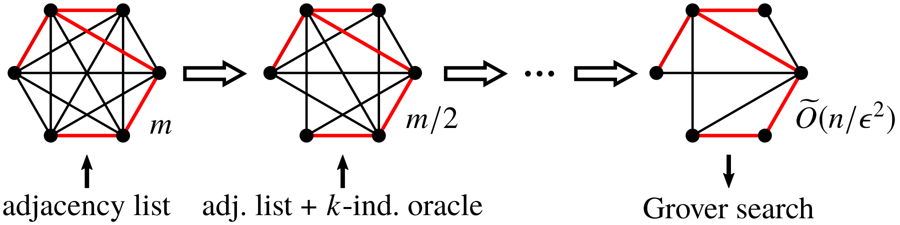
Quantum Sparsification Algorithm

1. $O(\log m)$ steps of quantum iterative sparsification, each requiring time $\widetilde{O}(\sqrt{mn}/\epsilon^2)$
2. use Grover search to retrieve $\widetilde{O}(n/\epsilon^2)$ surviving edges



Quantum Sparsification Algorithm

1. $O(\log m)$ steps of quantum iterative sparsification, each requiring time $\widetilde{O}(\sqrt{mn}/\epsilon^2)$
2. use Grover search to retrieve $\widetilde{O}(n/\epsilon^2)$ surviving edges



Theorem (this work)

There is a quantum algorithm that constructs an ϵ-cut sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges in time

$$\widetilde{O}(\sqrt{mn}/\epsilon^2)$$

Refined Quantum Sparsification Algorithm

we can improve ϵ-dependency in the runtime:

Refined Quantum Sparsification Algorithm

we can improve ϵ-dependency in the runtime:

1. create rough ϵ-spectral sparsifier H for $\epsilon = 1/10$
   $\rightarrow$ time $\widetilde{O}(\sqrt{mn})$ using our former algorithm

Refined Quantum Sparsification Algorithm

we can improve ϵ-dependency in the runtime:

1. create rough ϵ-spectral sparsifier H for $\epsilon = 1/10$

   $\rightarrow$ time $\widetilde{O}(\sqrt{mn})$ using our former algorithm
2. use Spielman-Srivastava to estimate effective resistances for H

   $\rightarrow$ time $\widetilde{O}(n)$ using Laplacian solving

Refined Quantum Sparsification Algorithm

we can improve ϵ-dependency in the runtime:

1. create rough ϵ-spectral sparsifier H for $\epsilon = 1/10$
   $\rightarrow$ time $\widetilde{O}(\sqrt{mn})$ using our former algorithm
2. use Spielman-Srivastava to estimate effective resistances for H
   $\rightarrow$ time $\widetilde{O}(n)$ using Laplacian solving
3. sample $\widetilde{O}(n/\epsilon^2)$ edges from G using these estimates
   $\rightarrow$ time $\widetilde{O}(\sqrt{mn/\epsilon^2})$ using quantum sampling

Refined Quantum Sparsification Algorithm

we can improve ϵ-dependency in the runtime:

1. create rough ϵ-spectral sparsifier H for $\epsilon = 1/10$

   $\rightarrow$ time $\widetilde{O}(\sqrt{mn})$ using our former algorithm
2. use Spielman-Srivastava to estimate effective resistances for H

   $\rightarrow$ time $\widetilde{O}(n)$ using Laplacian solving
3. sample $\widetilde{O}(n/\epsilon^2)$ edges from G using these estimates

   $\rightarrow$ time $\widetilde{O}(\sqrt{mn/\epsilon^2})$ using quantum sampling

Theorem (this work)

There is a quantum algorithm that constructs an ϵ-cut sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges in time

$$\widetilde{O}(\sqrt{mn}/\epsilon)$$

this work:

1. quantum algorithm to find H in time

$$\widetilde{O}(\sqrt{mn}/\epsilon)$$

2. matching $\widetilde{\Omega}(\sqrt{mn}/\epsilon)$ lower bound
3. quantum speedup (roughly $\widetilde{O}(m)$ to $\widetilde{O}(\sqrt{mn})$) for
   - max cut, min cut, min st-cut, sparsest cut, . . .
   - Laplacian solving, approximating resistances and RW properties, spectral clustering, . . .

Matching Lower Bound

intuition:

finding N marked elements among M elements takes time

$$\Omega(\sqrt{MN})$$

Matching Lower Bound

intuition:

finding N marked elements among M elements takes time

$$\Omega(\sqrt{MN})$$

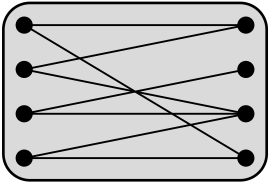
$\rightarrow$ finding $\widetilde{O}(n/\epsilon^2)$ edges of sparsifier among m edges takes time

$$\Omega(\sqrt{mn}/\epsilon)$$

Unsparsifiable Graph

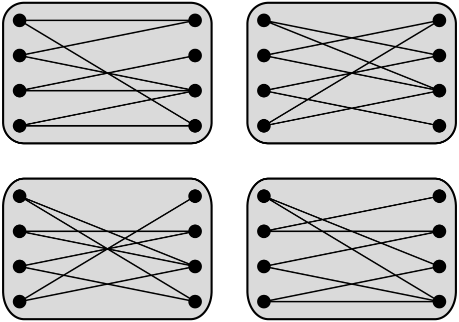
consider random graph B_ϵ

- bipartite with $1/\epsilon^2$ nodes left and right
- every left node connected to random subset of $1/(2\epsilon^2)$ right nodes



Unsparsifiable Graph

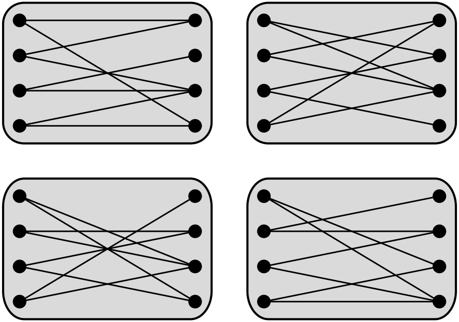
let $H(n, \epsilon)$ consist of $\epsilon^2 n/2$ random copies of B_ϵ



$H(n, \epsilon)$ has n nodes and $O(n/\epsilon^2)$ edges

Unsparsifiable Graph

let $H(n, \epsilon)$ consist of $\epsilon^2 n/2$ random copies of B_ϵ



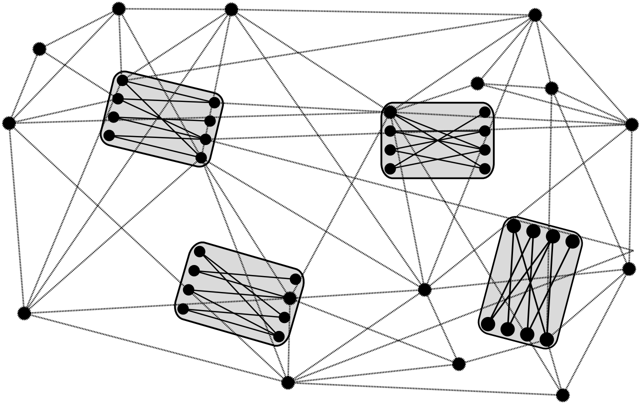
$H(n, \epsilon)$ has n nodes and $O(n/\epsilon^2)$ edges

Theorem (Andoni et al. '16)

With large probability, any ϵ-cut sparsifier of $H(n, \epsilon)$ must contain constant fraction of edges

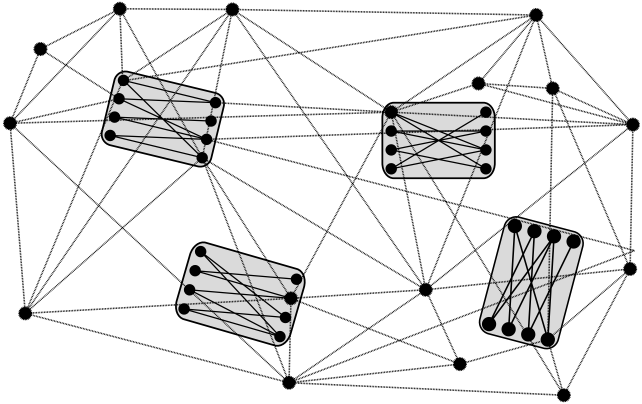
Hiding a Sparsifier

"hide" $H(n, \epsilon)$ in larger $G(n, m, \epsilon)$ with $O(n)$ nodes and $O(m)$ edges



Hiding a Sparsifier

"hide" $H(n, \epsilon)$ in larger $G(n, m, \epsilon)$ with $O(n)$ nodes and $O(m)$ edges



unit weight to edges in $H(n, \epsilon)$, others zero weight

$\rightarrow$ ϵ-spectral sparsifier of $G(n, m, \epsilon)$ must find constant fraction of $H(n, \epsilon)$

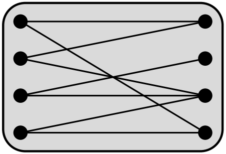
Proving a Lower Bound

adjacency list $\sim$ bit string $x \in \{0, 1\}^m$ with structure!

Proving a Lower Bound

adjacency list $\sim$ bit string $x \in \{0,1\}^m$ with structure!

single copy of B_ϵ:

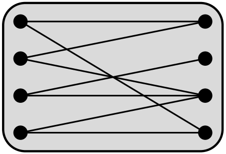


$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Proving a Lower Bound

adjacency list $\sim$ bit string $x \in \{0,1\}^m$ with structure!

single copy of B_ϵ:



$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

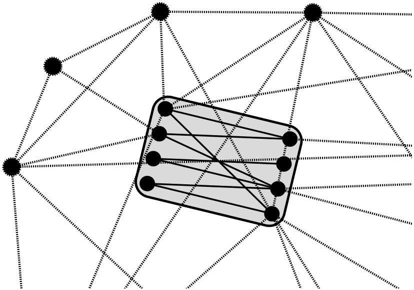
output constant fraction of 1-bits
= "relational problem" with complexity $\widetilde{\Omega}(1/\epsilon^2)$

$\rightarrow$ complexity for solving $\epsilon^2 n$ copies = $\widetilde{\Omega}(n)$

Proving a Lower Bound

"hidden" copy of B_ϵ:

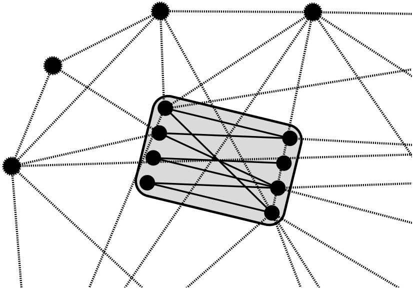
every entry of adjacency list is hidden among $N = m/(n\epsilon^2)$ entries



$$OR\left(\begin{bmatrix} {\scriptstyle 000000}1{\scriptstyle 000} & {\scriptstyle 0}0{\scriptstyle 00000000} & {\scriptstyle 0000000}0{\scriptstyle 00} & {\scriptstyle 00}1{\scriptstyle 0000000} \\ {\scriptstyle 000}1{\scriptstyle 000000} & {\scriptstyle 000000000}0 & {\scriptstyle 00000000}1{\scriptstyle 0} & {\scriptstyle 000}0{\scriptstyle 000000} \\ {\scriptstyle 00000}0{\scriptstyle 0000} & {\scriptstyle 000000}1{\scriptstyle 000} & {\scriptstyle 00000000}1{\scriptstyle 0} & {\scriptstyle 00000}0{\scriptstyle 0000} \\ 0{\scriptstyle 000000000} & {\scriptstyle 00}0{\scriptstyle 0000000} & {\scriptstyle 00000}1{\scriptstyle 0000} & {\scriptstyle 0000}1{\scriptstyle 00000} \end{bmatrix}\right) = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Proving a Lower Bound

"hidden" copy of B_ϵ:
every entry of adjacency list is hidden among $N = m/(n\epsilon^2)$ entries



$$OR\left(\begin{bmatrix} {\scriptstyle 000000}1{\scriptstyle 000} & {\scriptstyle 0}0{\scriptstyle 00000000} & {\scriptstyle 0000000}0{\scriptstyle 00} & {\scriptstyle 00}1{\scriptstyle 0000000} \\ {\scriptstyle 000}1{\scriptstyle 000000} & {\scriptstyle 000000000}0 & {\scriptstyle 00000000}1{\scriptstyle 0} & {\scriptstyle 000}0{\scriptstyle 000000} \\ {\scriptstyle 00000}0{\scriptstyle 0000} & {\scriptstyle 000000}1{\scriptstyle 000} & {\scriptstyle 00000000}1{\scriptstyle 0} & {\scriptstyle 00000}0{\scriptstyle 0000} \\ 0{\scriptstyle 000000000} & {\scriptstyle 00}0{\scriptstyle 0000000} & {\scriptstyle 00000}1{\scriptstyle 0000} & {\scriptstyle 0000}1{\scriptstyle 00000} \end{bmatrix}\right) = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

output constant fraction 1-bits, each described by OR_N-function
= relational problem composed with OR_N

Proving a Lower Bound

? quantum lower bound for composition relational problem and OR_N-function ($\Omega(\sqrt{N})$?

Proving a Lower Bound

? quantum lower bound for composition relational problem and OR_N-function ($\Omega(\sqrt{N})$?

Theorem (proof by A.Belovs and T.Lee, independently)

The query complexity of an efficiently verifiable relational problem, with lower bound L, composed with the OR_N-function, is $\Omega(L\sqrt{N})$.

Proving a Lower Bound

? quantum lower bound for composition relational problem and OR_N-function ($\Omega(\sqrt{N})$) ?

Theorem (proof by A.Belovs and T.Lee, independently)

The query complexity of an efficiently verifiable relational problem, with lower bound L, composed with the OR_N-function, is $\Omega(L\sqrt{N})$.

for $L = n$ and $N = m/(n\epsilon^2)$:

Corollary

The query complexity of explicity outputting an ϵ-spectral sparsifier of a graph with n nodes and m edges is

$$\widetilde{\Omega}(\sqrt{mn}/\epsilon).$$

this work:

1. quantum algorithm to find H in time

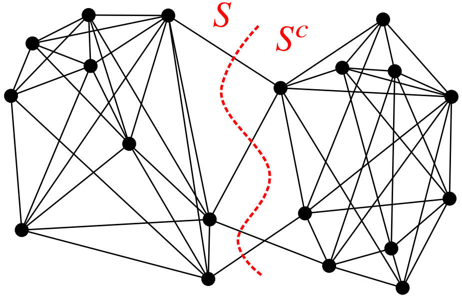
$$\widetilde{O}(\sqrt{mn}/\epsilon)$$

2. matching $\widetilde{\Omega}(\sqrt{mn}/\epsilon)$ lower bound
3. quantum speedup (roughly $\widetilde{O}(m)$ to $\widetilde{O}(\sqrt{mn})$) for
   - max cut, min cut, min st-cut, sparsest cut, ...
   - Laplacian solving, approximating resistances and RW properties, spectral clustering, ...

Cut Approximation

MIN CUT:

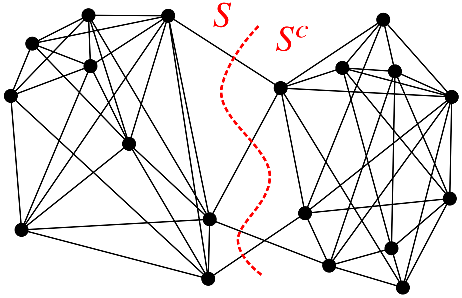
find cut (S, S^c) that minimizes $\mathrm{val}_G(S)$



Cut Approximation

MIN CUT:

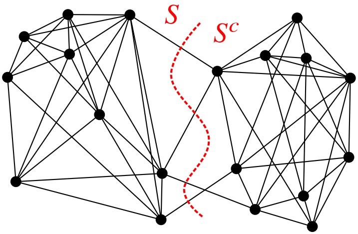
find cut (S, S^c) that minimizes $\mathrm{val}_G(S)$



classically: can find MIN CUT in time $\widetilde{O}(m)$ (Karger '00)

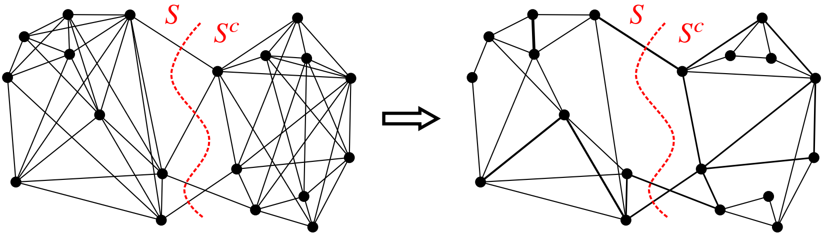
Cut Approximation

MIN CUT of ϵ-cut sparsifier H
gives ϵ-approximation of MIN CUT of G



Cut Approximation

MIN CUT of ϵ-cut sparsifier H
gives ϵ-approximation of MIN CUT of G



quantum sparsification ($\widetilde{O}(\sqrt{mn}/\epsilon)$) + classical MIN CUT on H ($\widetilde{O}(n/\epsilon^2)$)

= $\widetilde{O}(\sqrt{mn}/\epsilon)$ algorithm for ϵ-MIN CUT

Cut Approximation

	Classical	Quantum (this work)
.878-MAX CUT	$\widetilde{O}(m)$ (Arora-Kale'16)	$\widetilde{O}(\sqrt{mn})$
ϵ-MIN CUT	$\widetilde{O}(m)$ (Karger'00)	$\widetilde{O}(\sqrt{mn}/\epsilon)$
ϵ-MIN st-CUT	$\widetilde{O}(m + n/\epsilon^5)$ (Peng'16)	$\widetilde{O}(\sqrt{mn}/\epsilon + n/\epsilon^5)$
$\sqrt{\log n}$-SP. CUT/ -BAL. SEPARATOR	$\widetilde{O}(m + n^{1+\delta})$ (Sherman'09)	$\widetilde{O}(\sqrt{mn} + n^{1+\delta})$

Laplacian Solving

general linear system

$$Ax = b$$

Laplacian Solving

general linear system

$$Ax = b$$

$\rightarrow$ given A and b, complexity of finding x is $\widetilde{O}(n^{\omega})$ for $\omega < 2.373$

Laplacian Solving

general linear system

$$Ax = b$$

$\rightarrow$ given A and b, complexity of finding x is $\widetilde{O}(n^{\omega})$ for $\omega < 2.373$

Laplacian system

$$Lx = b$$

Laplacian Solving

general linear system

$$Ax = b$$

$\rightarrow$ given A and b, complexity of finding x is $\widetilde{O}(n^{\omega})$ for $\omega < 2.373$

Laplacian system

$$Lx = b$$

$\rightarrow$ given L and b, complexity of finding x is near-linear $\widetilde{O}(m) \in \widetilde{O}(n^2)$
(Spielman-Teng '04)

approximate Laplacian system using spectral sparsifier:

Lemma

Consider Laplacian system $L_G x = b$. If H is ϵ-spectral sparsifier of G, and $\tilde{x}$ solution to $L_H x = b$, then

$$\|\tilde{x} - x\|_{L_G} \leq 2\epsilon \|x\|_{L_G},$$

where $\|v\|_{L_G} = \|L_G^{1/2} v\|$.

Laplacian Solving

approximate Laplacian system using spectral sparsifier:

Lemma

Consider Laplacian system $L_G x = b$. If H is ϵ-spectral sparsifier of G, and $\tilde{x}$ solution to $L_H x = b$, then

$$\|\tilde{x} - x\|_{L_G} \leq 2\epsilon \|x\|_{L_G},$$

where $\|v\|_{L_G} = \|L_G^{1/2} v\|$.

quantum sparsification ($\widetilde{O}(\sqrt{mn}/\epsilon)$) + Laplacian solver on H ($\widetilde{O}(n/\epsilon^2)$)

= $\widetilde{O}(\sqrt{mn}/\epsilon)$ quantum algorithm for Laplacian solving

	Classical	Quantum (this work)
ϵ-Laplacian Solving	$\widetilde{O}(m)$ (ST'04)	$\widetilde{O}(\sqrt{mn}/\epsilon)$
ϵ-Effective Resistance (single)	$\widetilde{O}(m)$	$\widetilde{O}(\sqrt{mn}/\epsilon)$ prior: $\widetilde{O}(n^{3/2}/\epsilon^{3/2})$
ϵ-Effective Resistance (all)	$\widetilde{O}(m+n/\epsilon^4)$ (Spielman-Srivastava'08)	$\widetilde{O}(\sqrt{mn}/\epsilon+n/\epsilon^4)$
$O(1)$-Cover Time	$\widetilde{O}(m)$ (Ding-Lee-Peres'10)	$\widetilde{O}(\sqrt{mn})$
k bottom eigenvalues	$\widetilde{O}(m+kn/\epsilon^2)$	$\widetilde{O}(\sqrt{mn}/\epsilon+kn/\epsilon^2)$ prior, $k=1$: $\widetilde{O}(n^2/\epsilon)$
Spectral k-means clustering	$\widetilde{O}(m+n\,\mathrm{poly}(k))$	$\widetilde{O}(\sqrt{mn}+n\,\mathrm{poly}(k))$

Open Questions

Open Questions

- complexity $\widetilde{O}(\sqrt{mn}/\epsilon)$ of quantum Laplacian solver has linear dependence on error ϵ. Can we improve this to logarithmic?

Open Questions

- complexity $\widetilde{O}(\sqrt{mn}/\epsilon)$ of quantum Laplacian solver has linear dependence on error ϵ. Can we improve this to logarithmic?
- Laplacian solvers led to breakthrough $\widetilde{O}(m)$ algorithms for max flow. Can we find quantum speedup for max flow?

Open Questions

- complexity $\widetilde{O}(\sqrt{mn}/\epsilon)$ of quantum Laplacian solver has linear dependence on error ϵ. Can we improve this to logarithmic?
- Laplacian solvers led to breakthrough $\widetilde{O}(m)$ algorithms for max flow. Can we find quantum speedup for max flow?
- *exact* min cut can be found classically in $\widetilde{O}(m)$, we find ϵ-approximate in $\widetilde{O}(\sqrt{mn}/\epsilon)$. Can we also find exact min cut?

Open Questions

- complexity $\widetilde{O}(\sqrt{mn}/\epsilon)$ of quantum Laplacian solver has linear dependence on error ϵ. Can we improve this to logarithmic?
- Laplacian solvers led to breakthrough $\widetilde{O}(m)$ algorithms for max flow. Can we find quantum speedup for max flow?
- *exact* min cut can be found classically in $\widetilde{O}(m)$, we find ϵ-approximate in $\widetilde{O}(\sqrt{mn}/\epsilon)$. Can we also find exact min cut?
- spectral sparsification extended to hypergraphs, sums of PSD matrices, streaming, ... and is related to spectral sketching and data regression. Quantum speedup for these problems?