Quantum complexity of minimum cut

Simon Apers

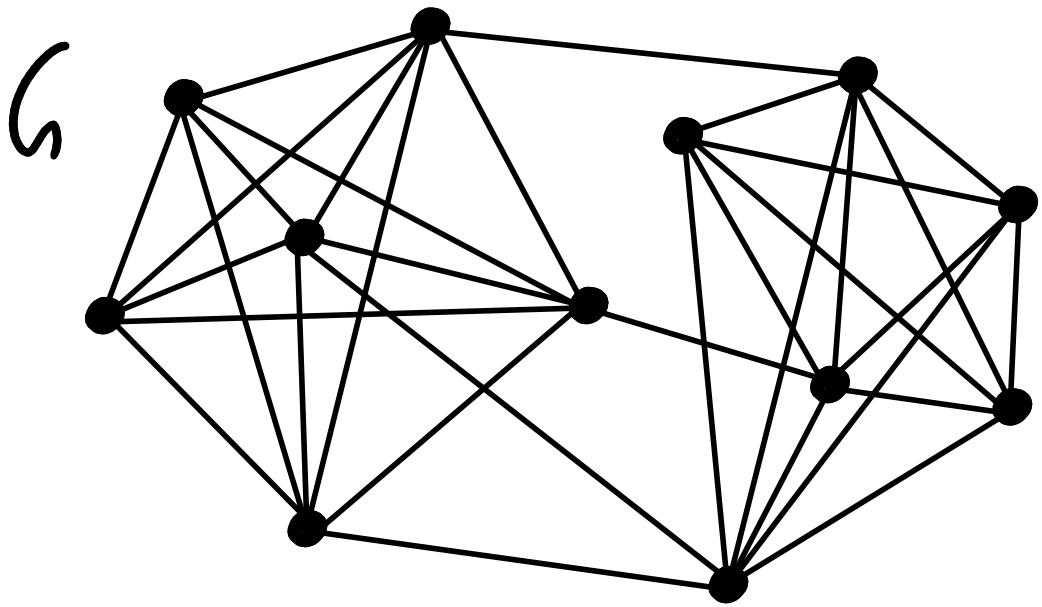
(CWI, ULB)

with Ronald de Wolf (FOCS '20, arXiv:1911.07306),
Troy Lee (CCC '21, arXiv:2011.09823) and Paweł Gawrychowski

Collège de France, June 2021

Graph

graph $G = (V, E, w)$

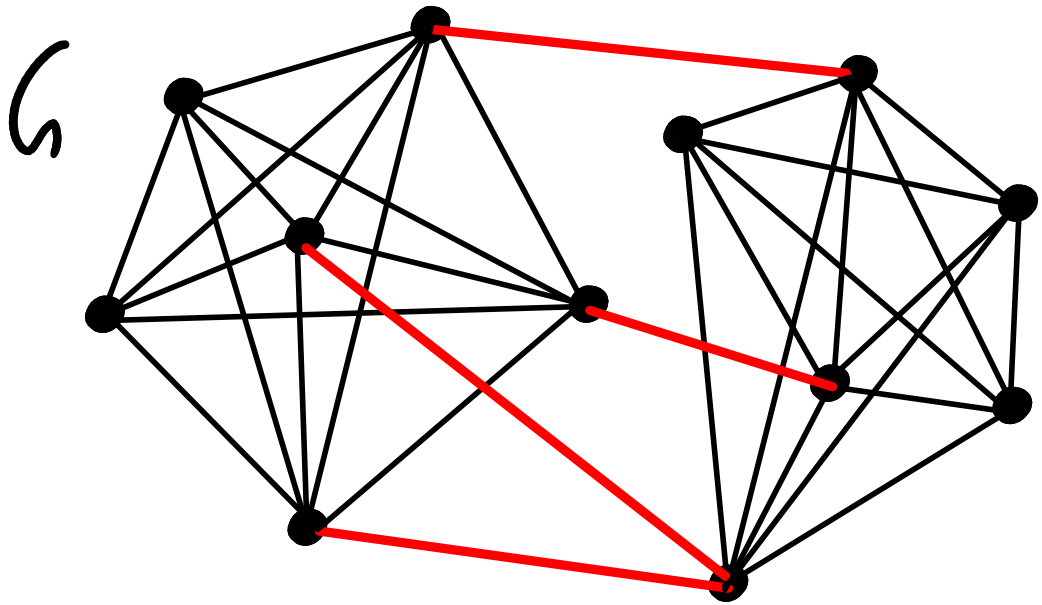


$|V| = n$ nodes, $|E| = m \leq \binom{n}{2}$ edges, weights w

Minimum cut

cut in G

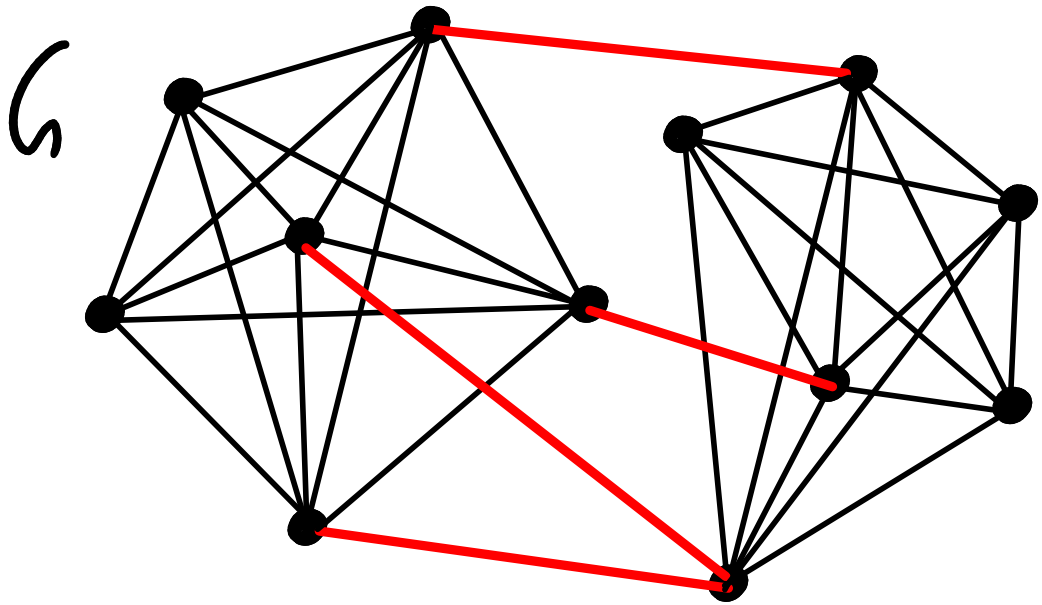
= set of edges $C \subseteq E$ such that $G - C$ is disconnected



Minimum cut

cut in G

= set of edges $C \subseteq E$ such that $G - C$ is disconnected



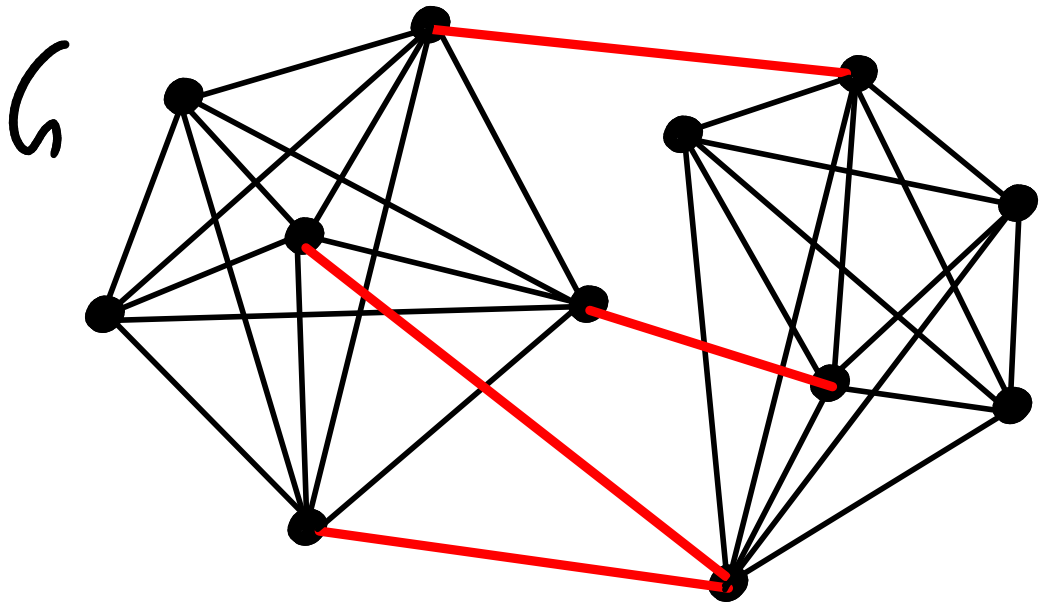
minimum cut (value) in G

= weight of cut with minimum total weight

Minimum cut

cut in G

= set of edges $C \subseteq E$ such that $G - C$ is disconnected



minimum cut (value) in G

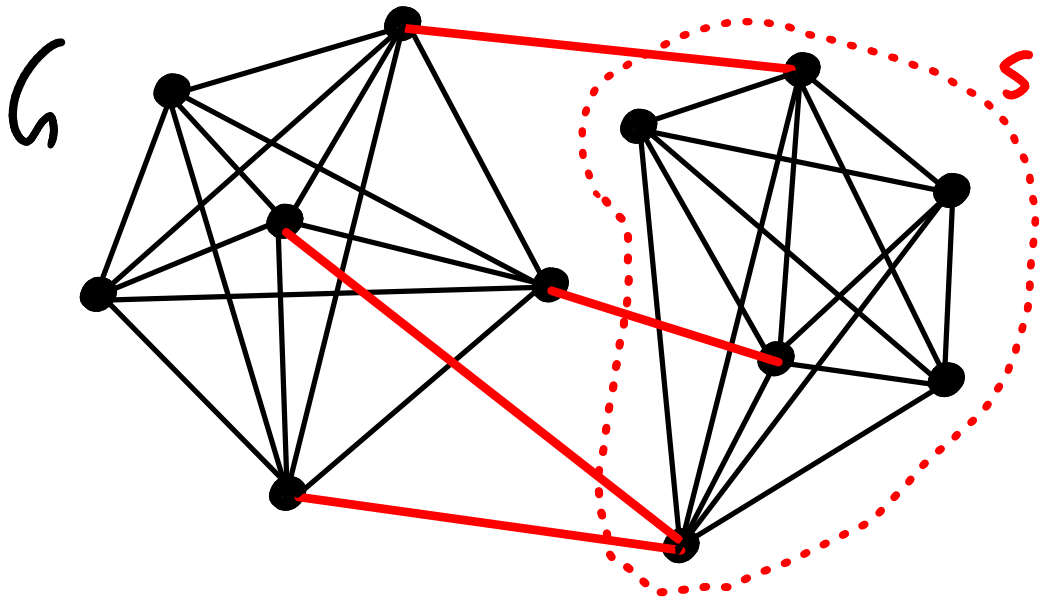
= weight of cut with minimum total weight

= $\min_{C \subseteq E}\{w(C) \mid G - C \text{ disconnected}\}$

Minimum cut

cut in G

= set of edges $C \subseteq E$ such that $G - C$ is disconnected



minimum cut (value) in G

= weight of cut with minimum total weight

= $\min_{C \subseteq E}\{w(C) \mid G - C \text{ disconnected}\}$

= $\min_{S \subset V}\{w(E(S, S^c))\}$

Minimum cut

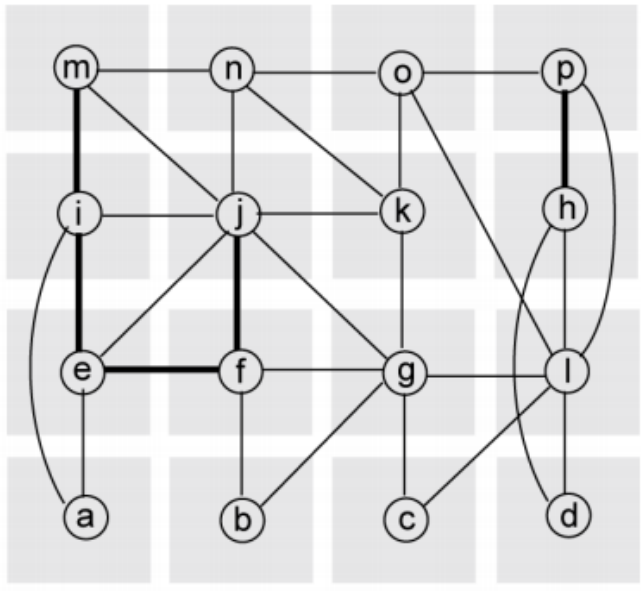
fundamental problem

in graph theory and combinatorial optimization

Minimum cut

fundamental problem
in graph theory and combinatorial optimization

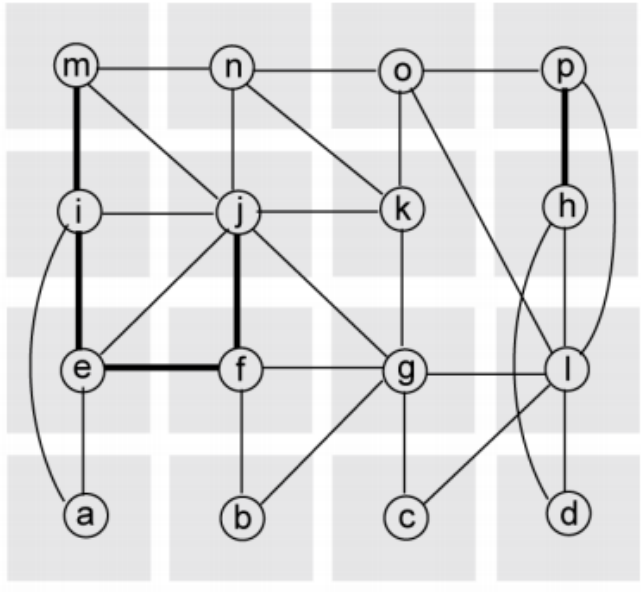
relevant for graph robustness,
graph partitioning (divide-and-conquer), VLSI design, . . .



Minimum cut

fundamental problem
in graph theory and combinatorial optimization

relevant for graph robustness,
graph partitioning (divide-and-conquer), VLSI design, . . .



related:
maximum flow, minimum k-cut,
minimum s-t cut, sparsest cut, . . .

Minimum cut

long line of research

from 1970s
(first polynomial algorithm by Podderyugin)

Minimum cut

long line of research

from 1970s
(first polynomial algorithm by Podderyugin)

through 1990s
(first near-linear time $\widetilde{O}(m)$ algorithm by Karger)

Minimum cut

long line of research

from 1970s
(first polynomial algorithm by Podderyugin)

through 1990s
(first near-linear time $\widetilde{O}(m)$ algorithm by Karger)

to today
(recent breakthrough by Li, arXiv:2106.05513))

Deterministic Mincut in Almost-Linear Time

Jason Li*
Carnegie Mellon University

December 16, 2020

Quantum complexity

this work:
quantum complexity of minimum cut

i.e., what is the complexity of solving
minimum cut on a quantum computer?

Quantum query complexity

quantum **query** complexity of minimum cut

= total number of queries to input

Quantum query complexity

quantum **query** complexity of minimum cut

= total number of queries to input

! often trivial for classical algorithms,
but sublinear for quantum algorithms

Quantum query complexity

quantum **query** complexity of minimum cut

= total number of queries to input

! often trivial for classical algorithms,
but sublinear for quantum algorithms

e.g., # queries to string $x \in \{0,1\}^N$ to determine $\mathrm{OR}(x)$?

classically = $\Omega(N)$, quantum = $O(\sqrt{N})$ (Grover search)

Quantum query complexity of graph problem

adjacency matrix of $G = (V, E, w)$:

$$A = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots \\ w_{1,2} & w_{2,2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \in \mathbb{R}^{n \times n}$$

input size $= n^2$

Quantum query complexity of graph problem

adjacency matrix of $G = (V, E, w)$:

$$A = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots \\ w_{1,2} & w_{2,2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \in \mathbb{R}^{n \times n}$$

input size $= n^2$

query:

$$|i, j, 0\rangle \mapsto |i, j, w_{i,j}\rangle$$

Quantum query complexity of graph problem

adjacency list of $G = (V, E, w)$:

(1): $(2, w_{1,2})$, $(6, w_{1,6})$, ...

(2): $(3, w_{2,3})$, ...

$\vdots$

(n): ...

input size $= m \leq \binom{n}{2}$

Quantum query complexity of graph problem

adjacency list of $G = (V, E, w)$:

(1): $(2, w_{1,2})$, $(6, w_{1,6})$, $\ldots$

(2): $(3, w_{2,3})$, $\ldots$

$\vdots$

(n): $\ldots$

input size $= m \leq \binom{n}{2}$

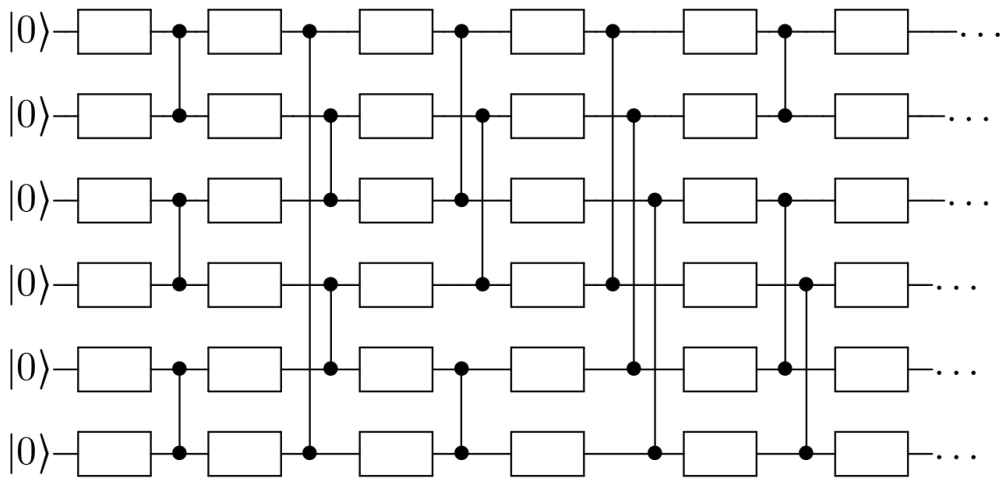
query:

$$|i, k, 0, 0\rangle \mapsto |i, k, n_k(i), w_{i,n_k(i)}\rangle$$

Quantum time complexity

quantum **time** complexity of minimum cut

= total number of elementary gates,
queries, QRAM operations*



* quantum-read/classical-write RAM

Motivation: quantum algorithms and graph problems

Why study quantum complexity of these problems?

Motivation: quantum algorithms and graph problems

Why study quantum complexity of these problems?

- long-term applications and
better understanding of quantum computing

Motivation: quantum algorithms and graph problems

Why study quantum complexity of these problems?

- long-term applications and
better understanding of quantum computing

- new insights in classical algorithms
(similar to streaming, dynamic, distributed, deterministic, . . . settings)

Motivation: quantum algorithms and graph problems

Why study quantum complexity of these problems?

- long-term applications and
better understanding of quantum computing

- new insights in classical algorithms
(similar to streaming, dynamic, distributed, deterministic, . . . settings)

- "true" complexity of a problem?

Motivation: quantum algorithms and graph problems

Why study quantum complexity of these problems?

- long-term applications and
better understanding of quantum computing

- new insights in classical algorithms
(similar to streaming, dynamic, distributed, deterministic, . . . settings)

- "true" complexity of a problem?

see e.g. talk "Quantum algorithms for optimization"
by Ronald de Wolf at Simons' institute ('21)

Quantum algorithms and graph problems

common caveats:

Quantum algorithms and graph problems

common caveats:

(subquadratic) polynomial speedup

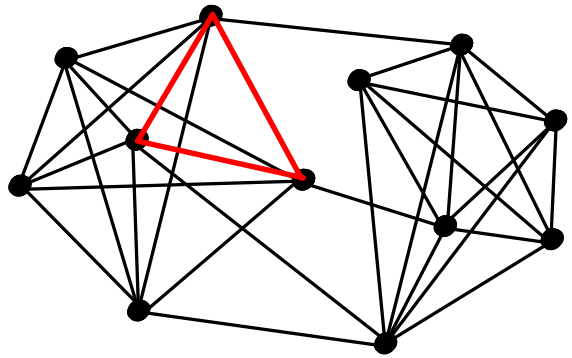
Quantum algorithms and graph problems

common caveats:

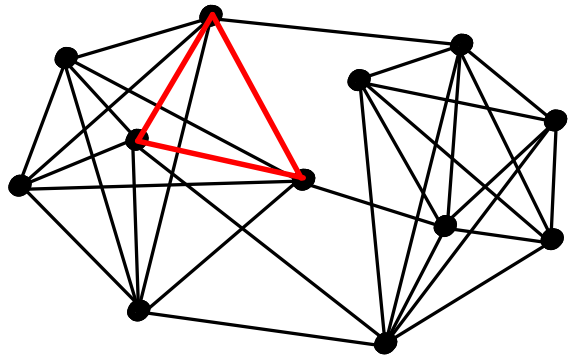
(subquadratic) polynomial speedup

QRAM requirements

Example: triangle finding



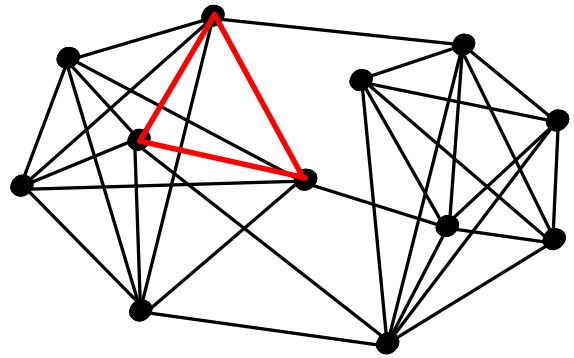
Example: triangle finding



- Grover search:

$O(n^{1.5})$, $\widetilde{O}(n^{1.428\ldots})$ ('03)

Example: triangle finding



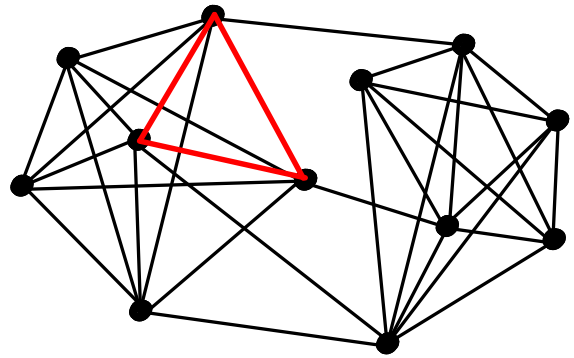
- Grover search:
$O(n^{1.5})$, $\widetilde{O}(n^{1.428\ldots})$ ('03)

- Grover search + quantum walks, learning graphs, . . . :
$\widetilde{O}(n^{1.3})$ ('05), $O(n^{1.296\ldots})$ ('12), $O(n^{1.285\ldots})$ ('13), $\widetilde{O}(n^{1.25})$ ('14)

* results by [Szegedy, '03], [Magniez-Santha-Szegedy, SICOMP'07], [Belovs, STOC'12],

[Lee-Magniez-Santha, SODA'13], [Le Gall, FOCS '14]

Example: triangle finding



- Grover search:
$O(n^{1.5})$, $\widetilde{O}(n^{1.428\ldots})$ ('03)

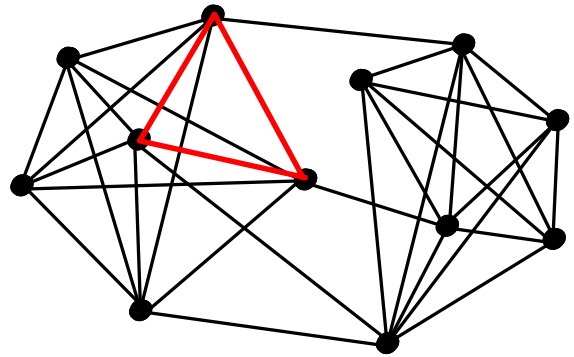
- Grover search + quantum walks, learning graphs, . . . :
$\widetilde{O}(n^{1.3})$ ('05), $O(n^{1.296\ldots})$ ('12), $O(n^{1.285\ldots})$ ('13), $\widetilde{O}(n^{1.25})$ ('14)

$\leftrightarrow \Omega(n)$ lower bound

* results by [Szegedy, '03], [Magniez-Santha-Szegedy, SICOMP'07], [Belovs, STOC'12],

[Lee-Magniez-Santha, SODA'13], [Le Gall, FOCS '14]

Example: triangle finding



- Grover search:
$O(n^{1.5})$, $\widetilde{O}(n^{1.428\ldots})$ ('03)

- Grover search + quantum walks, learning graphs, . . . :
$\widetilde{O}(n^{1.3})$ ('05), $O(n^{1.296\ldots})$ ('12), $O(n^{1.285\ldots})$ ('13), $\widetilde{O}(n^{1.25})$ ('14)

$\leftrightarrow \Omega(n)$ lower bound

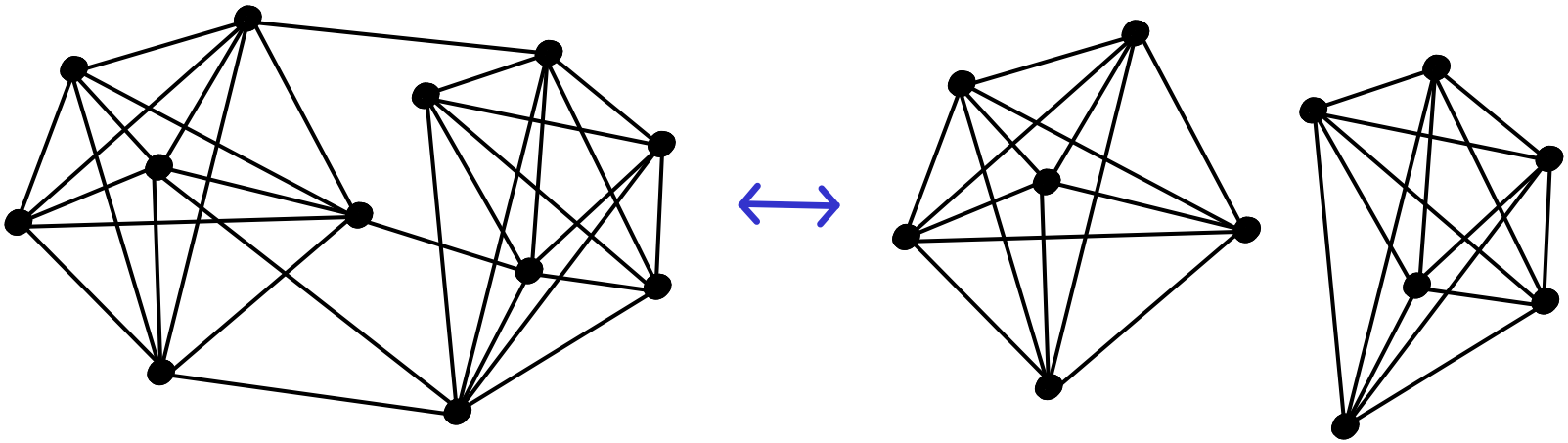
related to element distinctness
often paired with new tools in quantum query complexity

* results by [Szegedy, '03], [Magniez-Santha-Szegedy, SICOMP'07], [Belovs, STOC'12],

[Lee-Magniez-Santha, SODA'13], [Le Gall, FOCS '14]

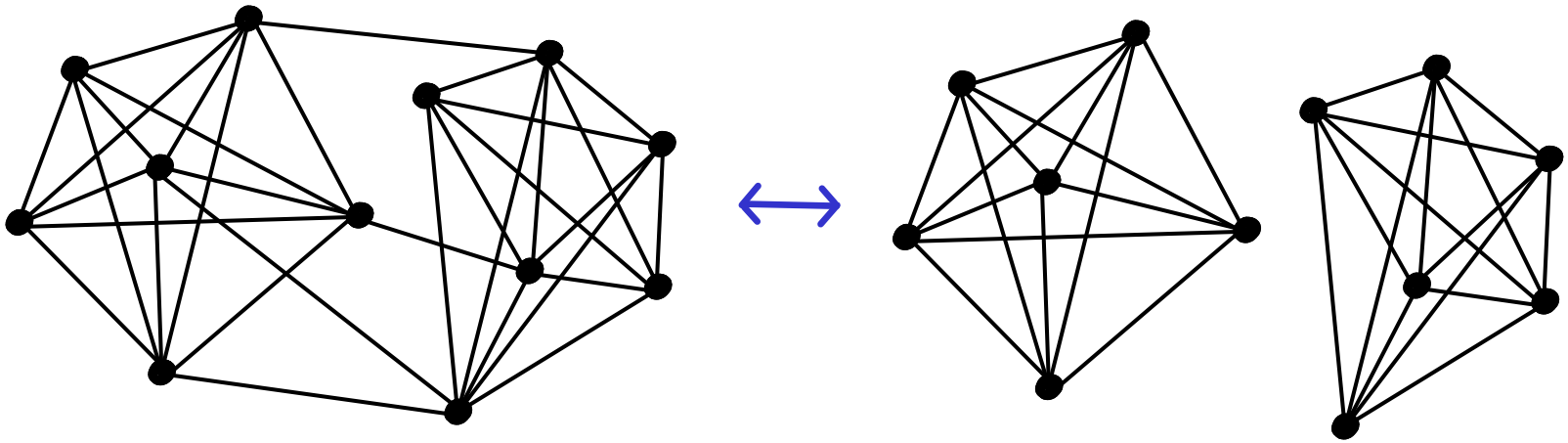
Example: connectivity

Is graph G connected?



Example: connectivity

Is graph G connected?



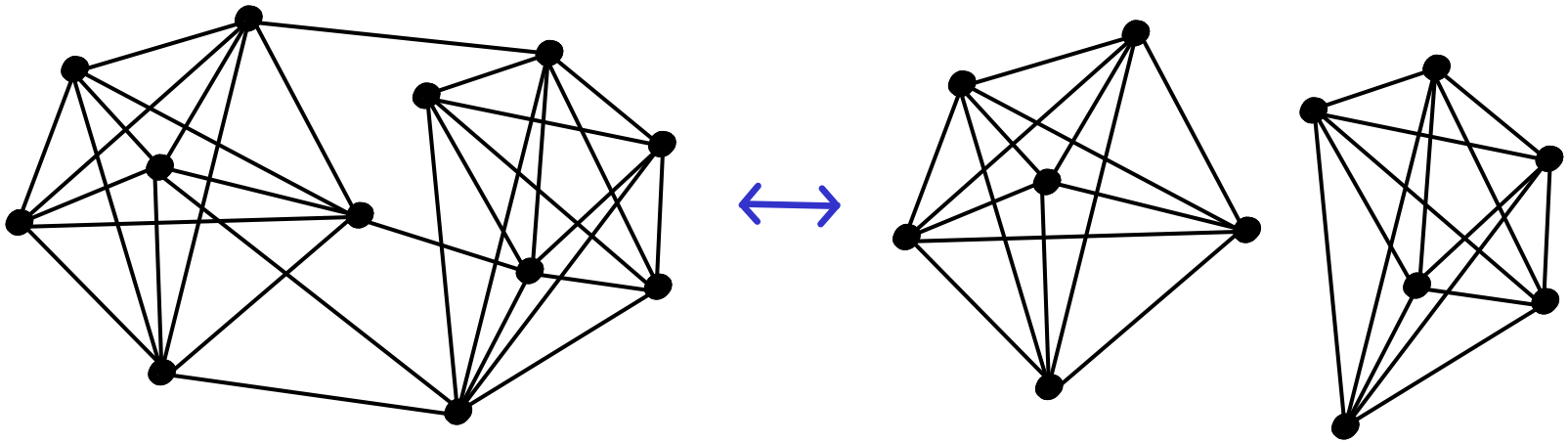
Theorem (Dürr-Heiligman-Høyer-Mhalla, SICOMP '06)

The quantum complexity of connectivity is

$$\widetilde{\Theta}(n^{3/2}) \text{ [M]} \quad \textit{and} \quad \widetilde{\Theta}(n) \text{ [L]}.$$

Example: connectivity

Is graph G connected?



Theorem (Dürr-Heiligman-Høyer-Mhalla, SICOMP '06)

The quantum complexity of connectivity is

$$\widetilde{\Theta}(n^{3/2}) \text{ [M]} \quad \textit{and} \quad \widetilde{\Theta}(n) \text{ [L]}.$$

$\rightarrow \Omega(n^{3/2})$ [M] and $\Omega(n)$ [L] for minimum cut!

Quantum query bounds for minimum cut

Limits on quantum speedups

quantum query complexity problem:

given quantum query access to

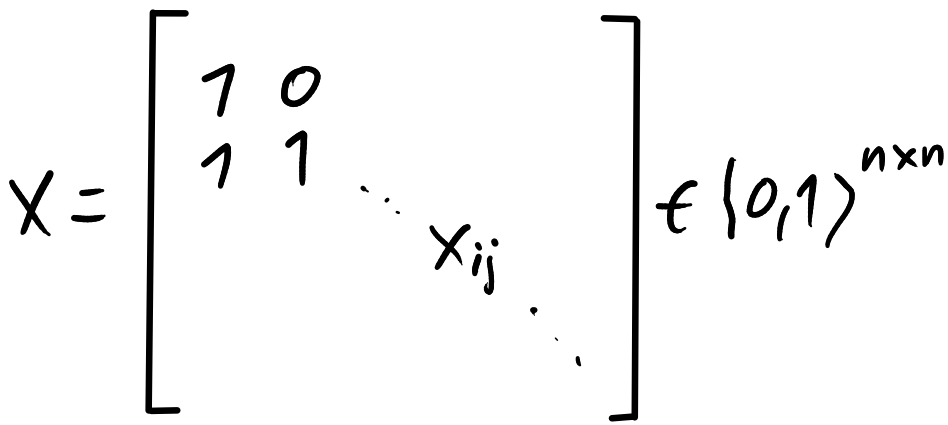
$$X = \begin{bmatrix} 1 & 0 & & \\ 1 & 1 & \ddots & \\ & & x_{ij} & \\ & & & \ddots \end{bmatrix} \in \{0,1\}^{n \times n}$$

determine Hamming weight $|x| = \sum x_{ij}$

Limits on quantum speedups

quantum query complexity problem:

given quantum query access to

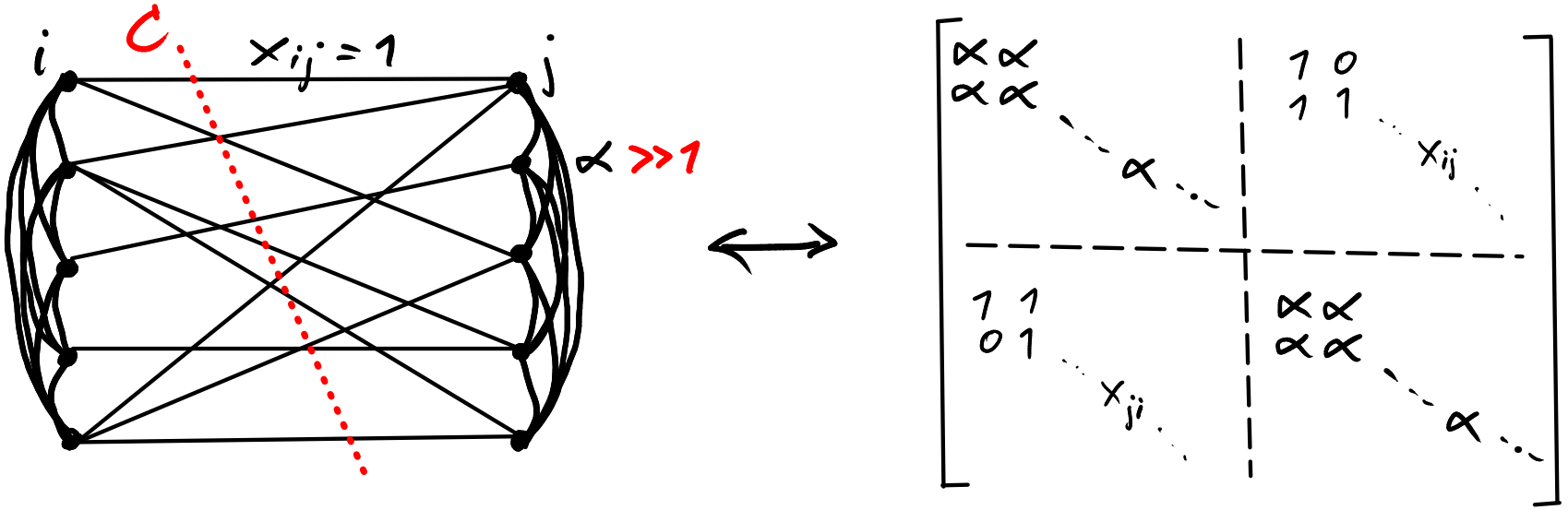


determine Hamming weight $|x| = \sum x_{ij}$

[Beals-Buhrman-Cleve-Mosca-de Wolf, FOCS '98]:
"counting" requires $n^2/2$ quantum queries to x
$\rightarrow$ no quantum speedup for counting

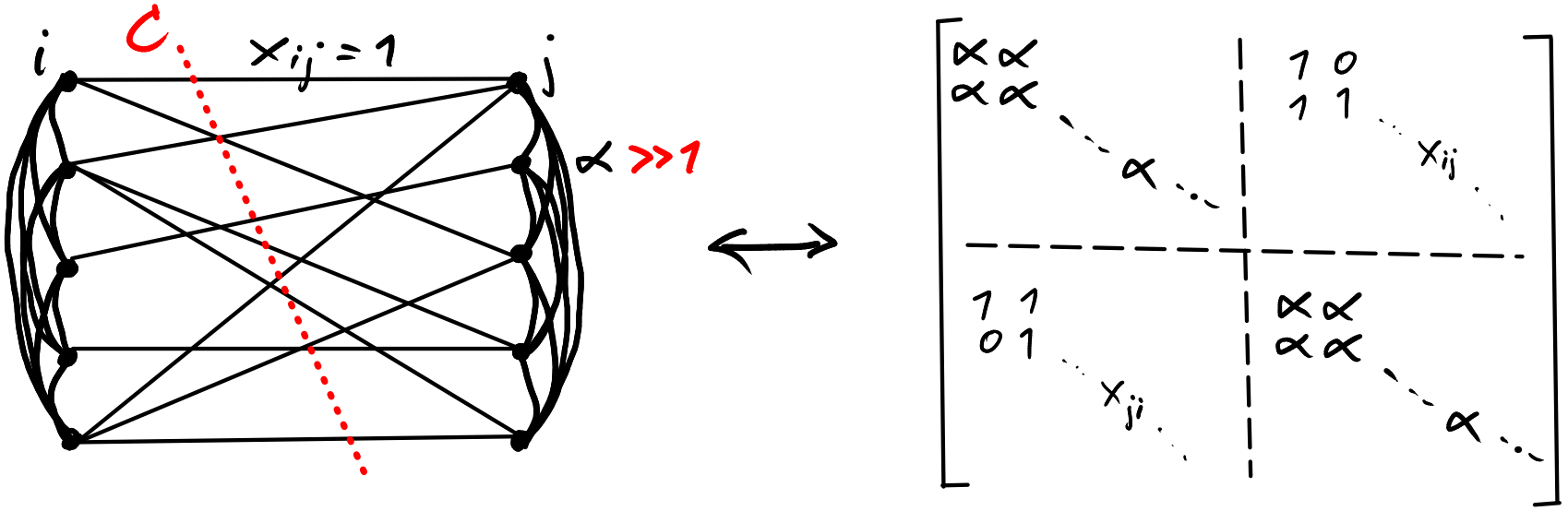
Limits on quantum speedups

minimum cut instance [M]:



Limits on quantum speedups

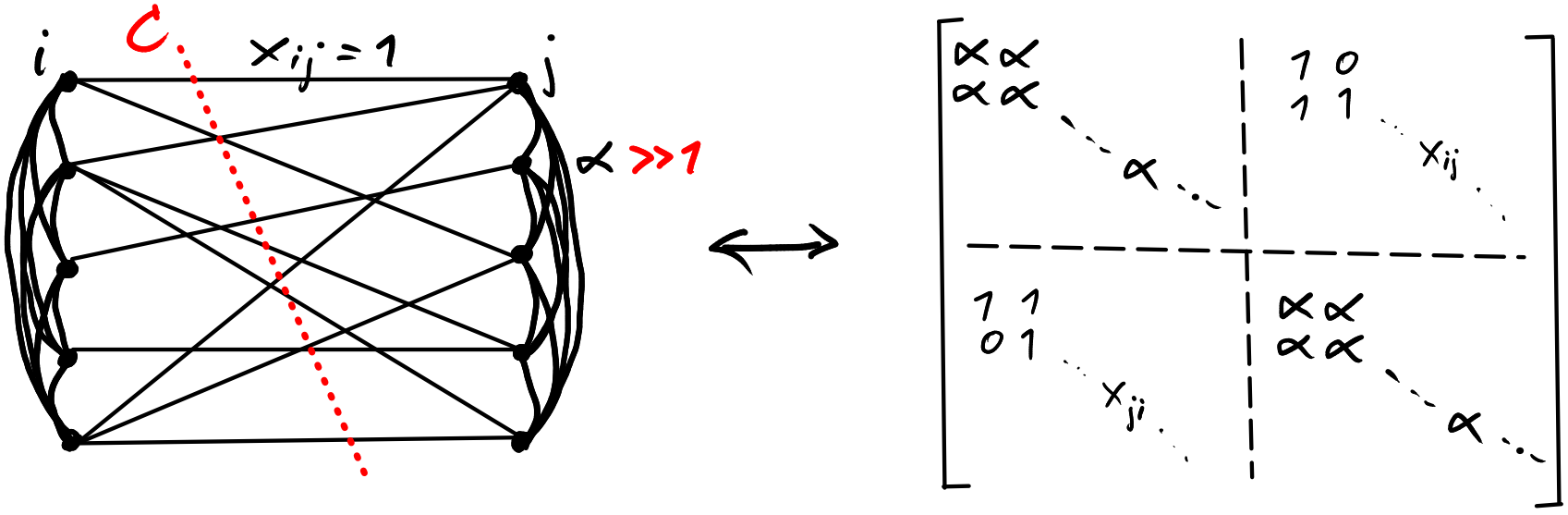
minimum cut instance [M]:



value minimum cut $w(C) = |x|$,
so counting reduces to minimum cut

Limits on quantum speedups

minimum cut instance [M]:



value minimum cut $w(C) = |x|$,
so counting reduces to minimum cut

↓

no quantum speedup for minimum cut!

Limits on quantum speedups

“approximate counting” loophole:

Limits on quantum speedups

"approximate counting" loophole:

quantum speedup for ϵ-approximating $|x|$
[Brassard-Høyer-Mosca-Tapp '00]

Limits on quantum speedups

"approximate counting" loophole:

quantum speedup for ϵ-approximating $|x|$
[Brassard-Høyer-Mosca-Tapp '00]

$\downarrow$

quantum speedup for *approximate* minimum cut?

Limits on quantum speedups

“small counting” loophole:

Limits on quantum speedups

"small counting" loophole:

quantum speedup for exact counting if $|x|$ is *small*

Limits on quantum speedups

"small counting" loophole:

quantum speedup for exact counting if $|x|$ is *small*

if $|x| \leq k$, then computing $|x|$ requires $O(\sqrt{n^2 k})$ quantum queries
(use Grover search to find all 1's in x)

Limits on quantum speedups

"small counting" loophole:

quantum speedup for exact counting if $|x|$ is *small*

if $|x| \leq k$, then computing $|x|$ requires $O(\sqrt{n^2 k})$ quantum queries
(use Grover search to find all 1's in x)

$\downarrow$

quantum speedup for minimum cut with few edges?

Quantum algorithm for *approximate* minimum cut

based on [A - de Wolf, FOCS '20]

key idea (exploits “approximate counting” loophole):

key idea (exploits “approximate counting” loophole):

1. quantum speedup for constructing “graph sparsifier”

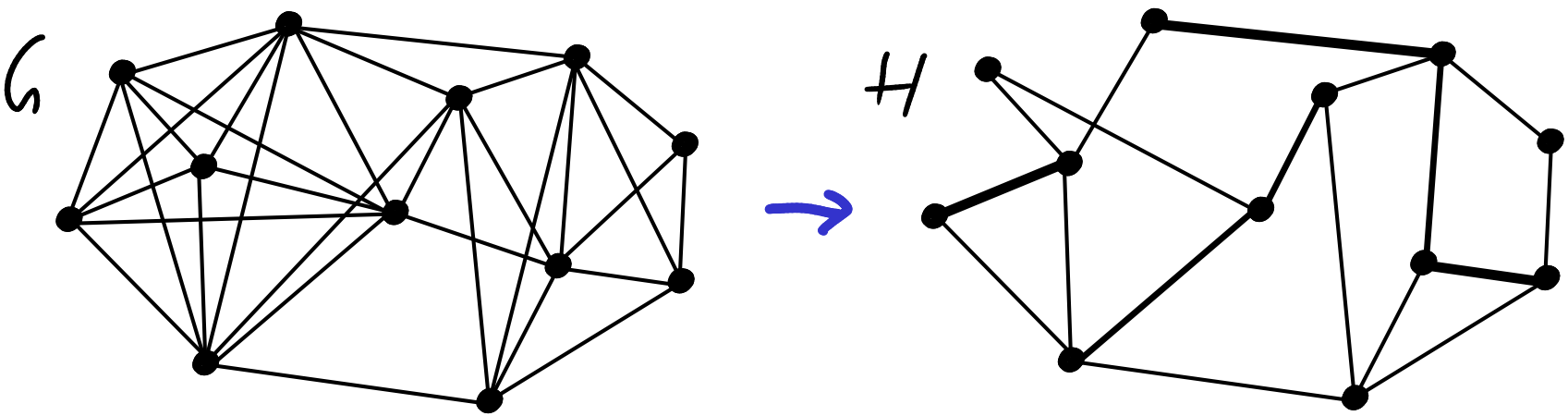
key idea (exploits “approximate counting” loophole):

1. quantum speedup for constructing “graph sparsifier”

2. solve minimum cut in sparsifier

Graph sparsification

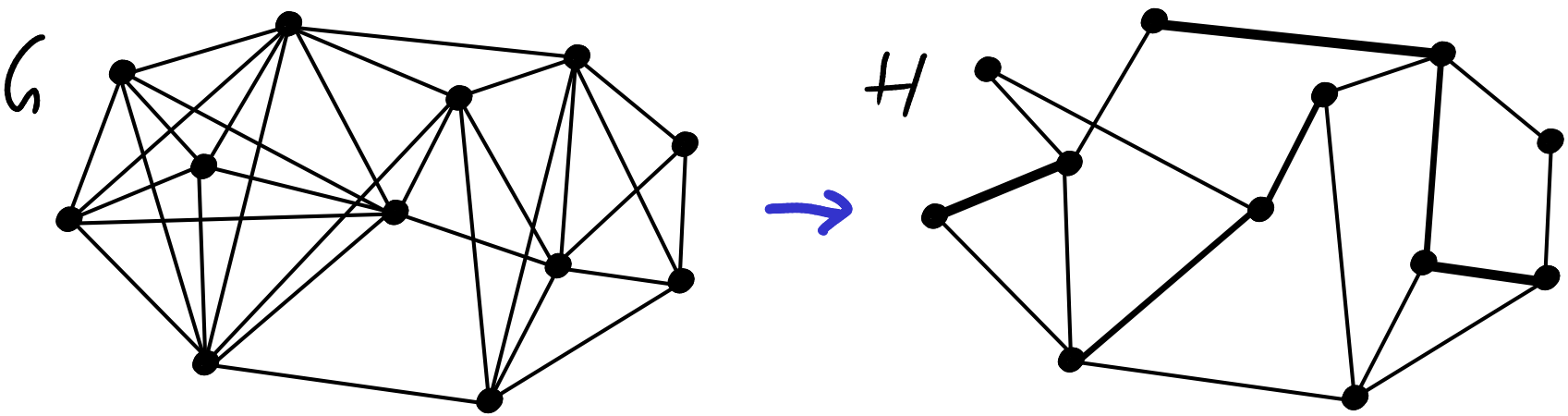
cut sparsifier H is subgraph of G such that



Graph sparsification

cut sparsifier H is subgraph of G such that

- H is sparse

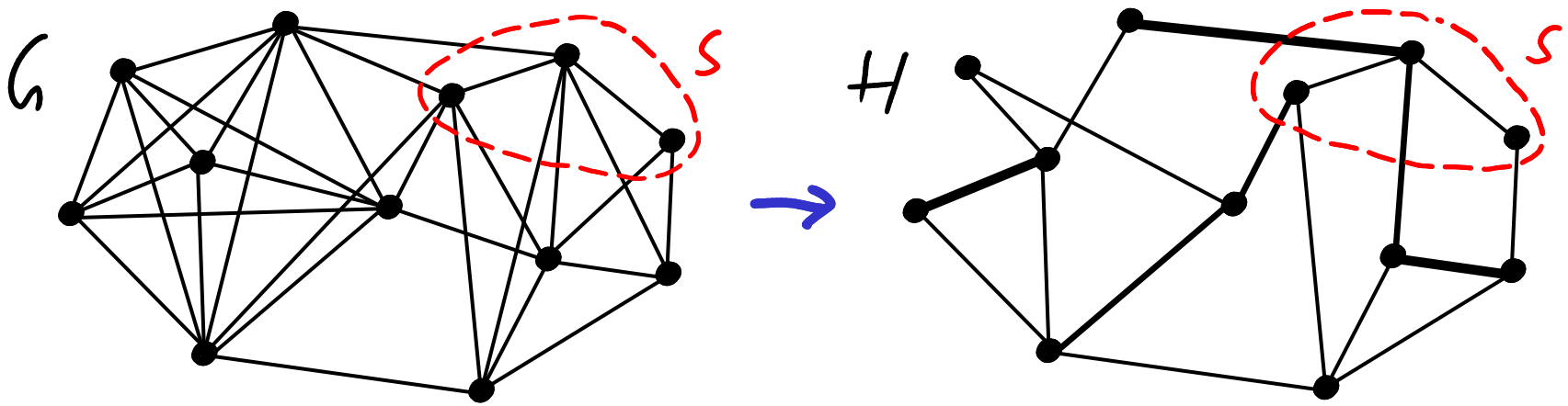


Graph sparsification

cut sparsifier H is subgraph of G such that

- H is sparse
- cuts in H approximate cuts in G:

$$w_H(E_H(S, S^c)) = (1 \pm \epsilon)\, w_G(E_G(S, S^c)), \quad \forall S \subset V$$

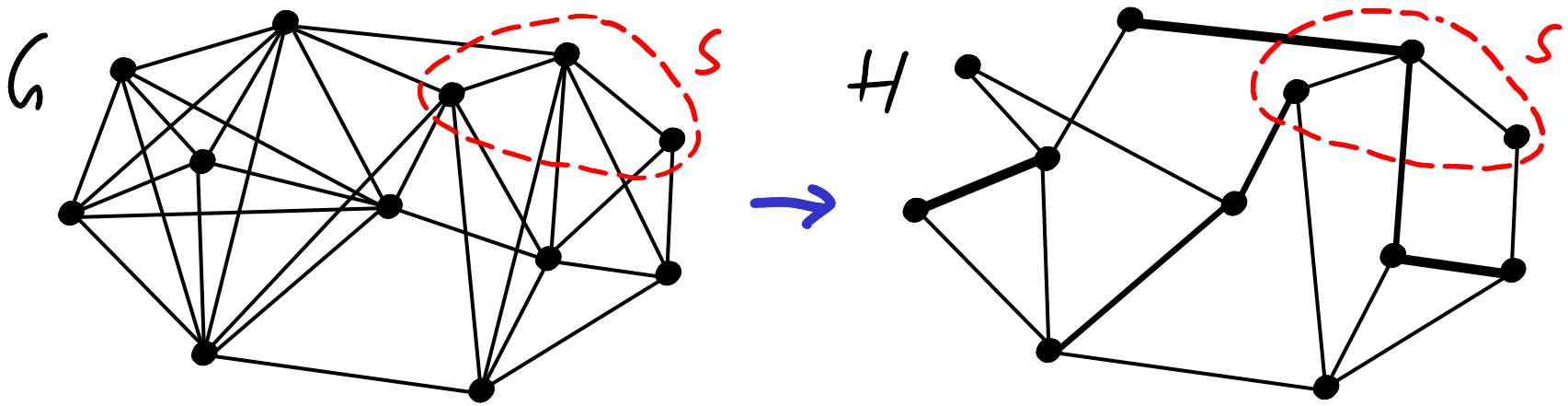


Graph sparsification

cut sparsifier H is subgraph of G such that

- H is sparse
- cuts in H approximate cuts in G:

$$w_H(E_H(S, S^c)) = (1 \pm \epsilon)\, w_G(E_G(S, S^c)), \quad \forall S \subset V$$



! in particular: minimum cut of H ϵ-approximates minimum cut of G

Graph sparsification

? how sparse can H be ?

Theorem (Karger '94, Benczúr-Karger '96)

Every graph G has an ϵ-cut sparsifier H with $\widetilde{O}(n/\epsilon^2)$ edges, which can be found in time $\widetilde{O}(m)$.

Graph sparsification

? how sparse can H be ?

Theorem (Karger '94, Benczúr-Karger '96)

Every graph G has an ϵ-cut sparsifier H with $\widetilde{O}(n/\epsilon^2)$ edges, which can be found in time $\widetilde{O}(m)$.

- building block of many classical approximation algorithms for graph problems

Graph sparsification

? how sparse can H be ?

Theorem (Karger '94, Benczúr-Karger '96)

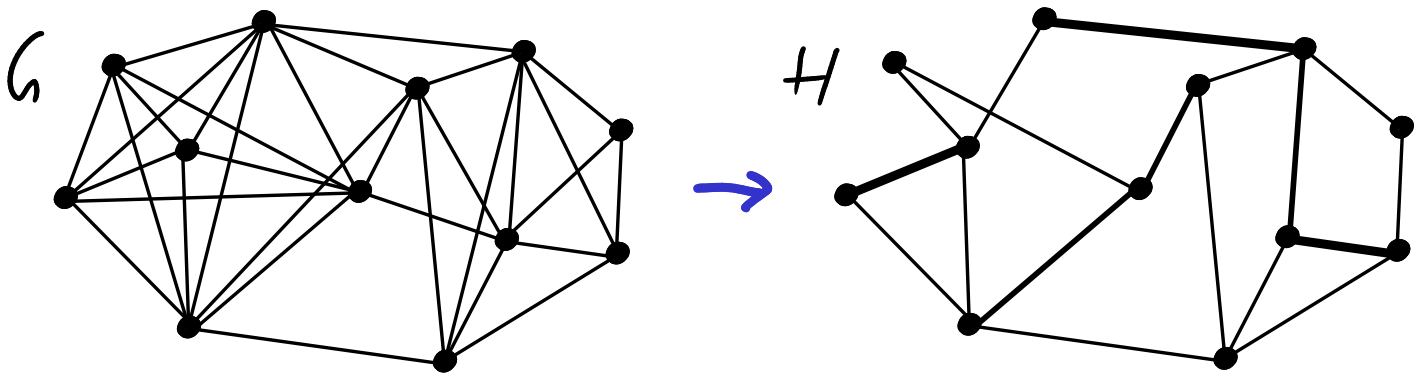
Every graph G has an ϵ-cut sparsifier H with $\widetilde{O}(n/\epsilon^2)$ edges, which can be found in time $\widetilde{O}(m)$.

- building block of many classical approximation algorithms for graph problems
- crucial component in efficient classical algorithms for *exact* minimum cut (Karger, J. ACM '00)

Quantum speedup for sparsification

Theorem (A-de Wolf '20)

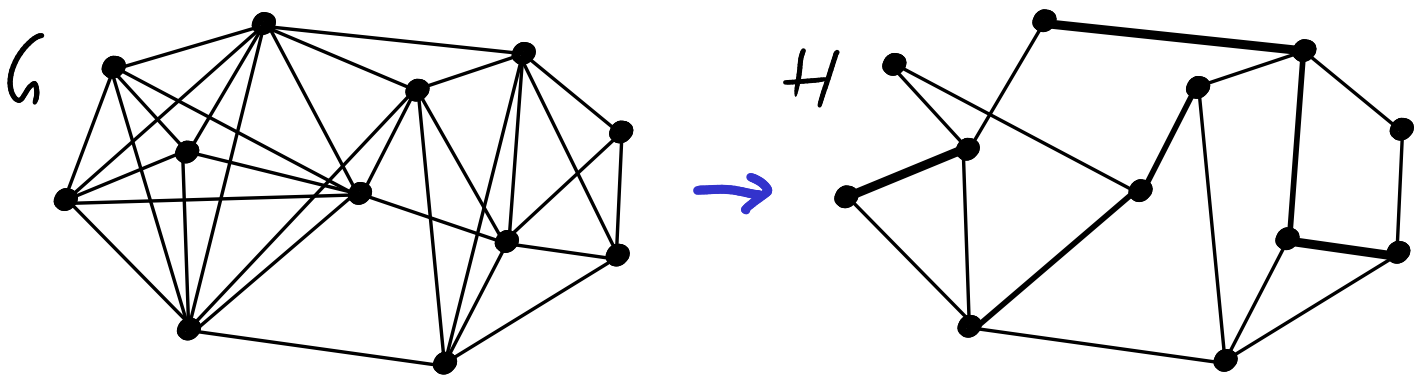
There is a quantum algorithm for constructing an ϵ-cut sparsifier H in time $\widetilde{O}(\sqrt{mn}/\epsilon)$ [L], which is optimal.



Quantum speedup for sparsification

Theorem (A-de Wolf '20)

There is a quantum algorithm for constructing an ϵ-cut sparsifier H in time $\widetilde{O}(\sqrt{mn}/\epsilon)$ [L], which is optimal.

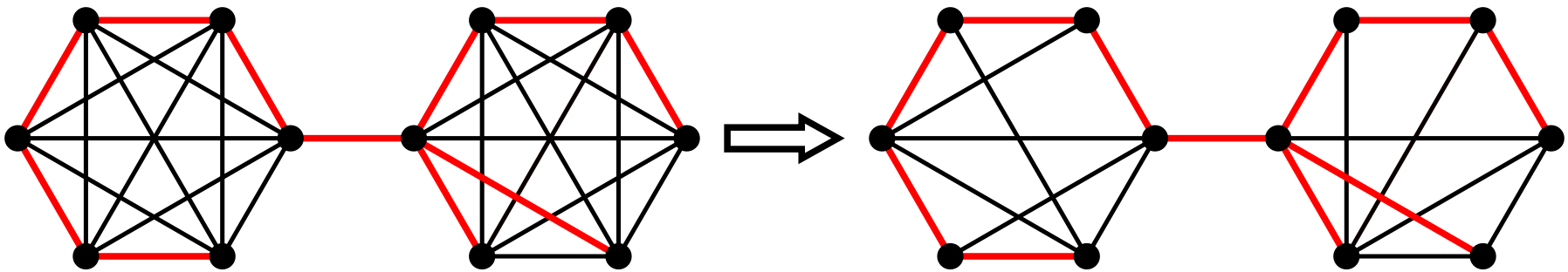


e.g., if $m \sim n^2$ then speedup from $\sim n^2$ (classical) to $\sim n^{3/2}$ (quantum)

Classical algorithm for sparsification

Iterative sparsification:

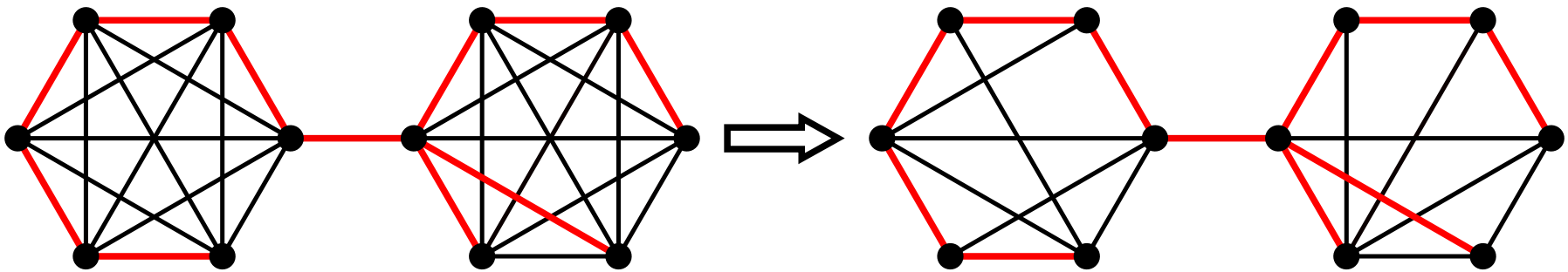
[Koutis-Xu ’16, Fung-Hariharan-Harvey-Panigrahi ’19]



Classical algorithm for sparsification

Iterative sparsification:

[Koutis-Xu '16, Fung-Hariharan-Harvey-Panigrahi '19]

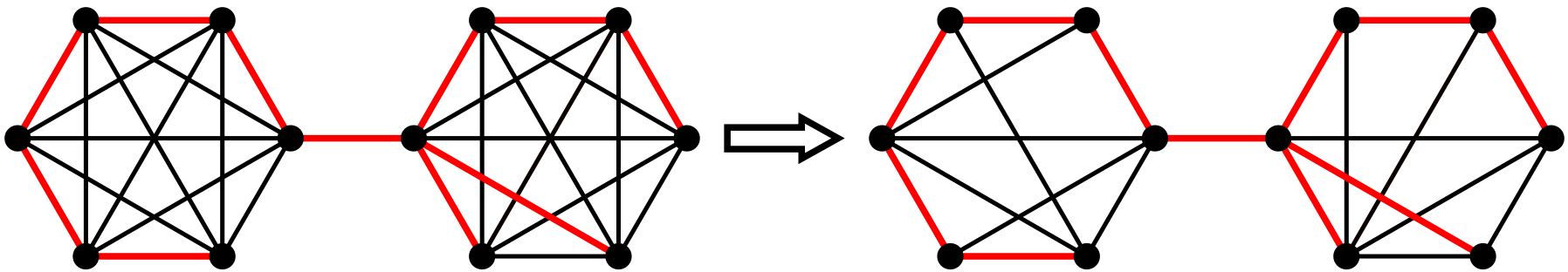


1. construct $\widetilde{O}(1/\epsilon^2)$ minimum spanning trees and keep these edges

Classical algorithm for sparsification

Iterative sparsification:

[Koutis-Xu '16, Fung-Hariharan-Harvey-Panigrahi '19]

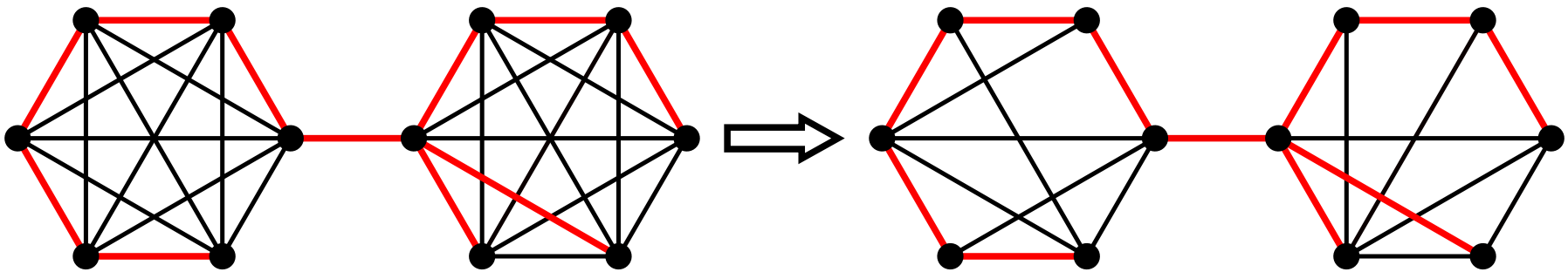


1. construct $\widetilde{O}(1/\epsilon^2)$ minimum spanning trees and keep these edges
2. keep any remaining edge with probability $1/2$

Classical algorithm for sparsification

Iterative sparsification:

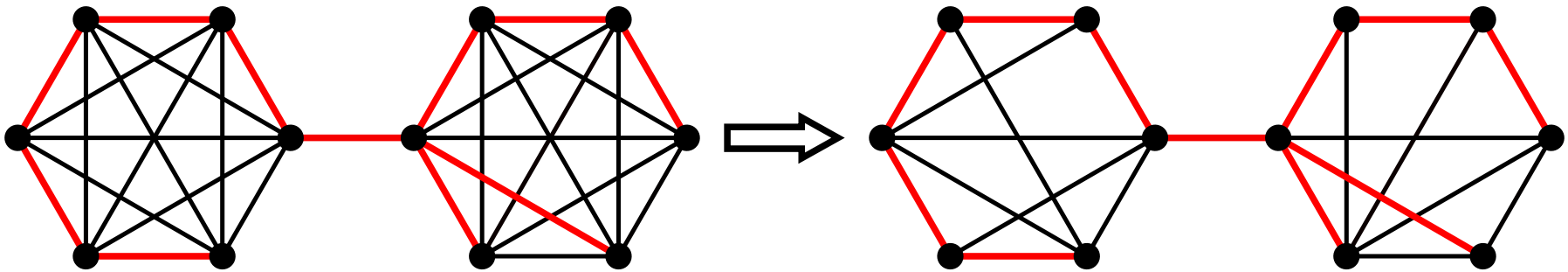
[Koutis-Xu '16, Fung-Hariharan-Harvey-Panigrahi '19]



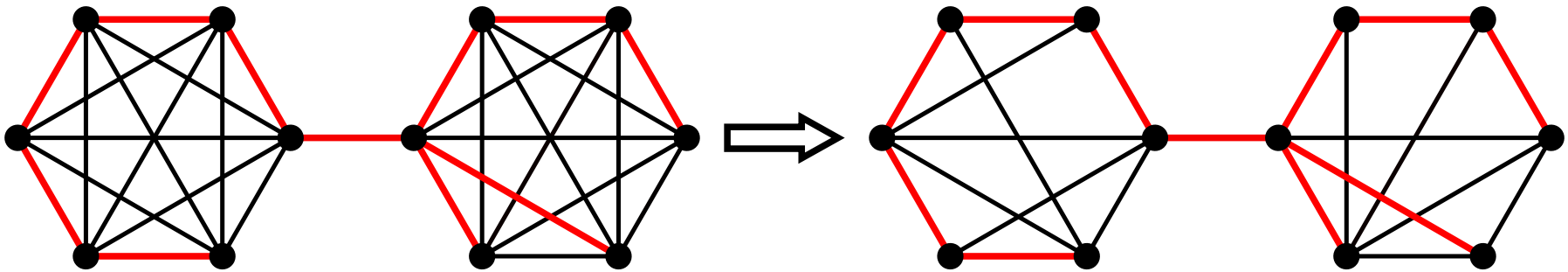
1. construct $\widetilde{O}(1/\epsilon^2)$ minimum spanning trees and keep these edges
2. keep any remaining edge with probability $1/2$

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

Quantum algorithm for sparsification

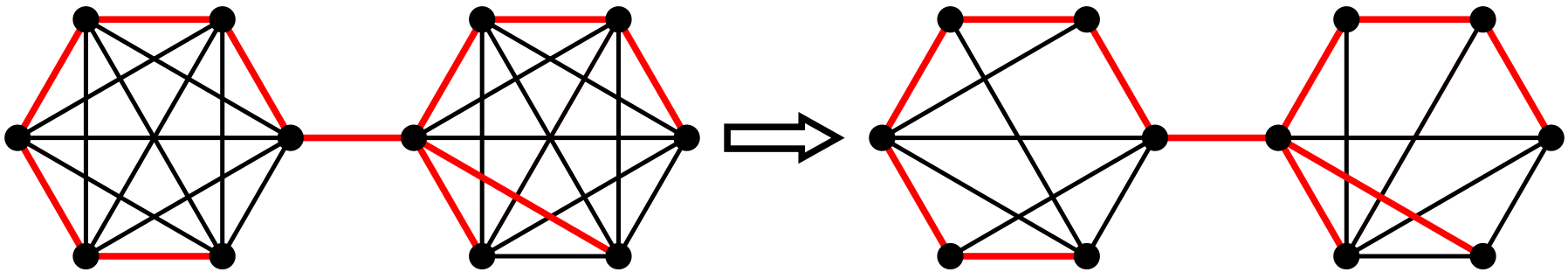


Quantum algorithm for sparsification



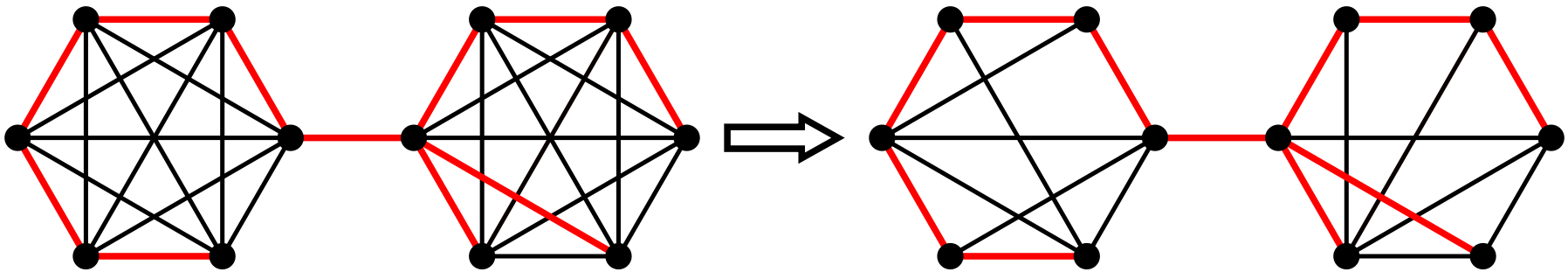
1. use quantum algorithm to construct $\widetilde{O}(1/\epsilon^2)$ minimum spanning trees and keep these edges

Quantum algorithm for sparsification



1. use quantum algorithm to construct $\widetilde{O}(1/\epsilon^2)$ minimum spanning trees and keep these edges
2. implicitly sample any remaining edge with probability $1/2$

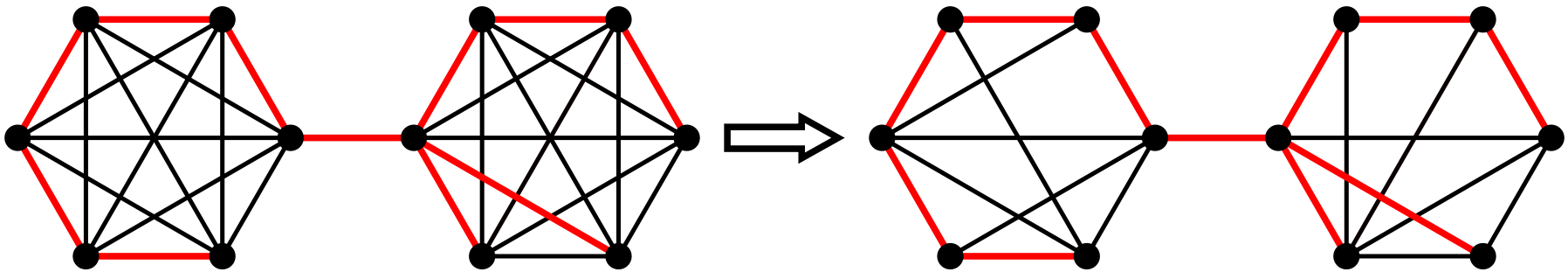
Quantum algorithm for sparsification



1. use quantum algorithm to construct $\widetilde{O}(1/\epsilon^2)$ minimum spanning trees and keep these edges
2. implicitly sample any remaining edge with probability $1/2$

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

Quantum algorithm for sparsification

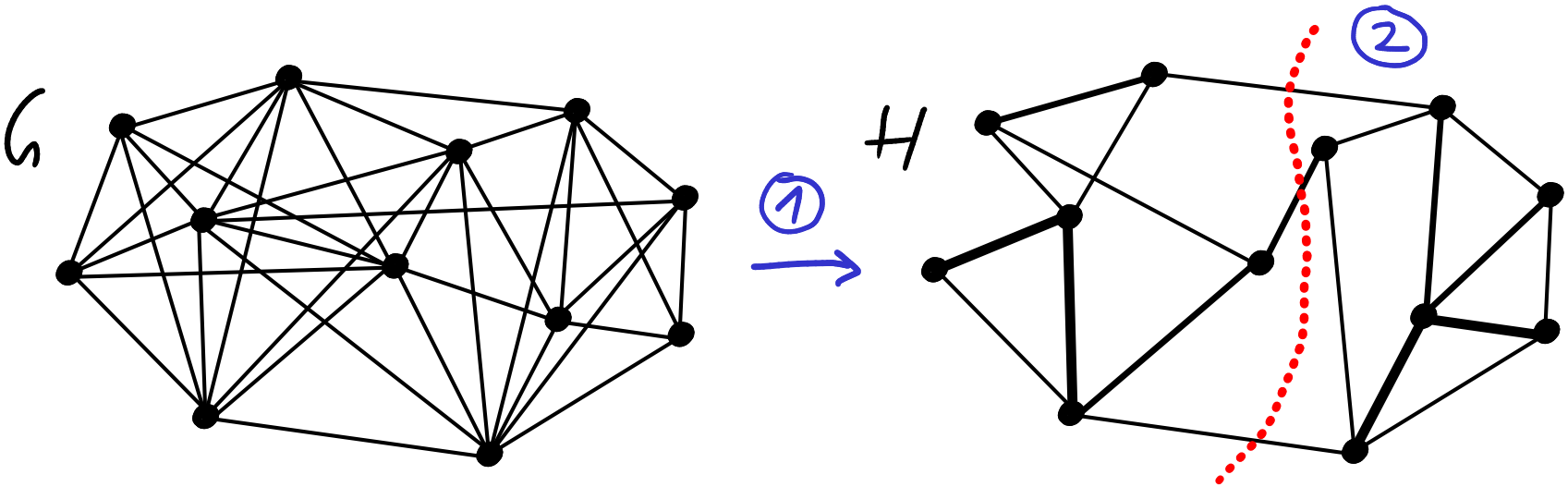


1. use quantum algorithm to construct $\widetilde{O}(1/\epsilon^2)$ minimum spanning trees and keep these edges
2. implicitly sample any remaining edge with probability $1/2$

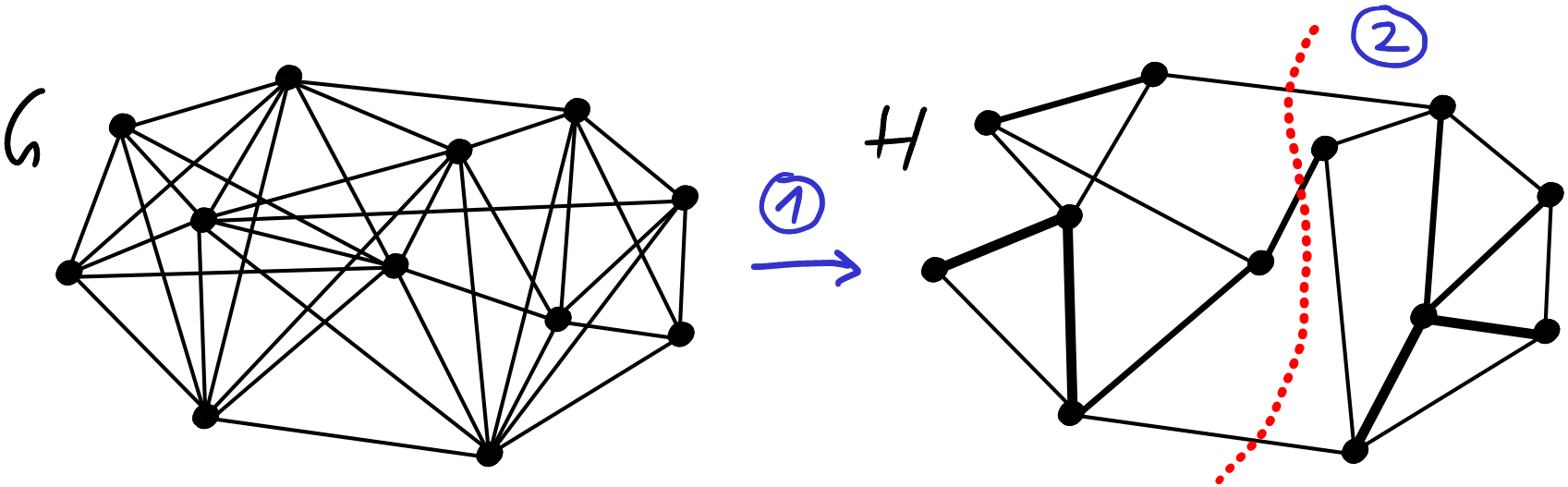
$\rightarrow$ *repeat* $O(\log n)$ *times:* ϵ-cut sparsifier with $\widetilde{O}(n/\epsilon^2)$ edges

total complexity (after more work): $\widetilde{O}(\sqrt{mn}/\epsilon)$

Quantum speedup for approximating minimum cut

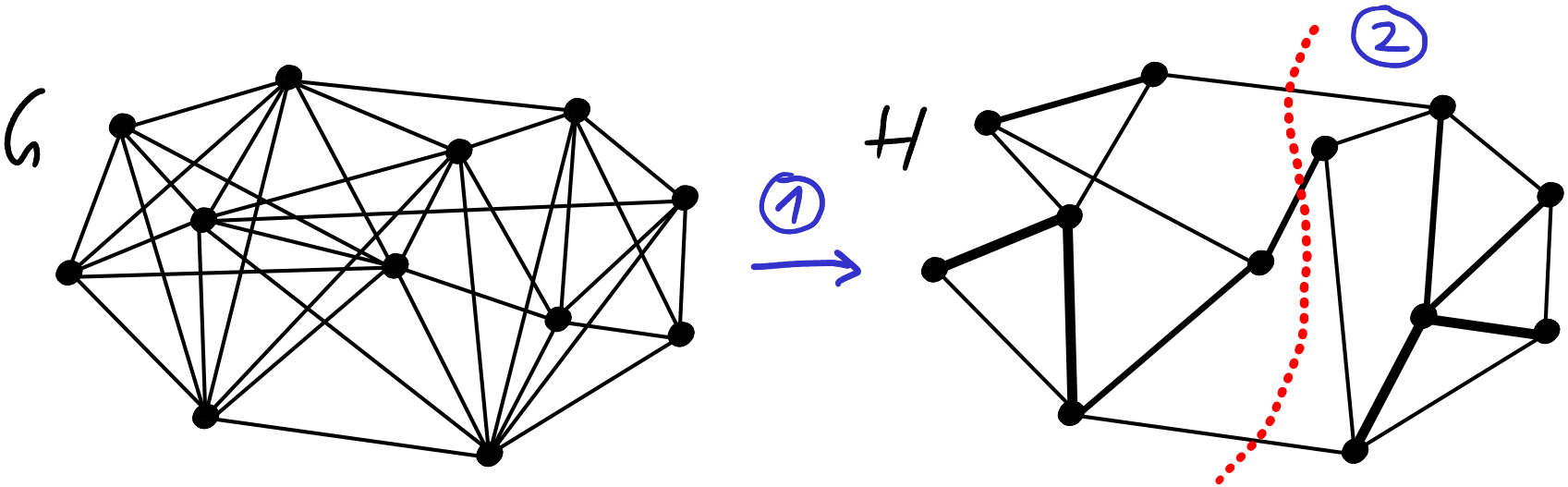


Quantum speedup for approximating minimum cut



1: Construct ϵ-sparsifier H ▷ $\widetilde{O}(\sqrt{mn}/\epsilon)$ queries/time

Quantum speedup for approximating minimum cut



1: Construct ϵ-sparsifier H ▷ $\widetilde{O}(\sqrt{mn}/\epsilon)$ queries/time

2: Solve minimum cut in H ▷ no queries, $\widetilde{O}(n)$ time

---

Quantum speedup for approximating minimum cut

Theorem

There is a quantum algorithm for ϵ-approximating the minimum cut of a weighted graph in time $\widetilde{O}(\sqrt{mn}/\epsilon)$.

Quantum speedup for approximating minimum cut

Theorem

There is a quantum algorithm for ϵ-approximating the minimum cut of a weighted graph in time $\widetilde{O}(\sqrt{mn}/\epsilon)$.

Similarly, quantum speedup for

- approximating other cut problems (max cut, sparsest cut, . . .)
- approximately solving Laplacian systems
- . . .

Quantum speedup for approximating minimum cut

Theorem

There is a quantum algorithm for ϵ-approximating the minimum cut of a weighted graph in time $\widetilde{O}(\sqrt{mn}/\epsilon)$.

Similarly, quantum speedup for

- approximating other cut problems (max cut, sparsest cut, ...)
- approximately solving Laplacian systems
- ...

Open question: is $\widetilde{O}(\sqrt{mn}/\epsilon)$ optimal?

Quantum algorithm for *exact* minimum cut in *simple* graphs

based on [A - Lee, CCC '21]

key idea (exploits “small counting” loophole):

key idea (exploits “small counting” loophole):

1. simple graphs have *small* number of edges in minimum cuts

key idea (exploits “small counting” loophole):

1. simple graphs have *small* number of edges in minimum cuts
2. find these edges using graph sparsifier and Grover search

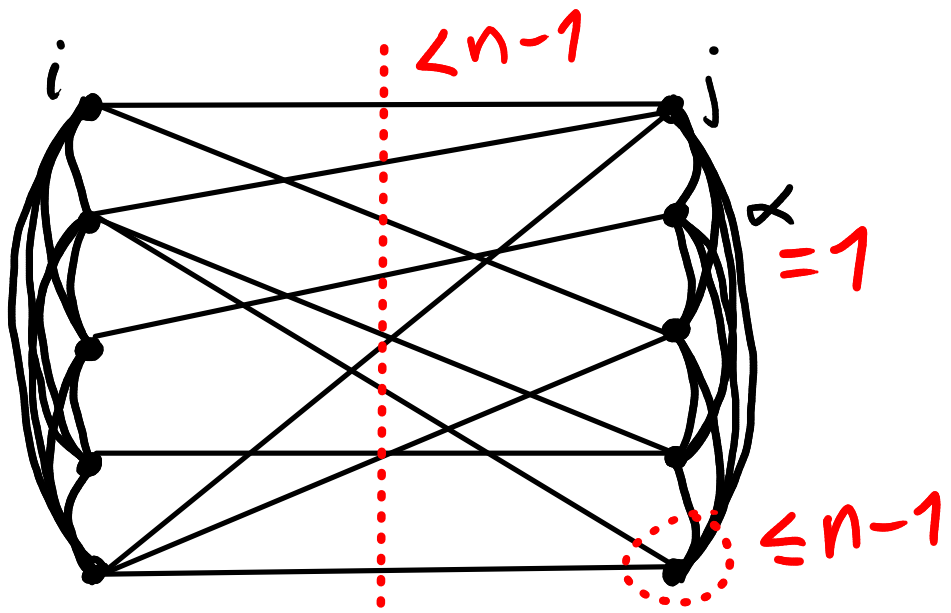
key idea (exploits “small counting” loophole):

1. simple graphs have *small* number of edges in minimum cuts

2. find these edges using graph sparsifier and Grover search

! first focus on query complexity

“simple graphs have *small* number of edges involved in minimum cuts”
[Kawarabayashi-Thorup, J. ACM ’18]

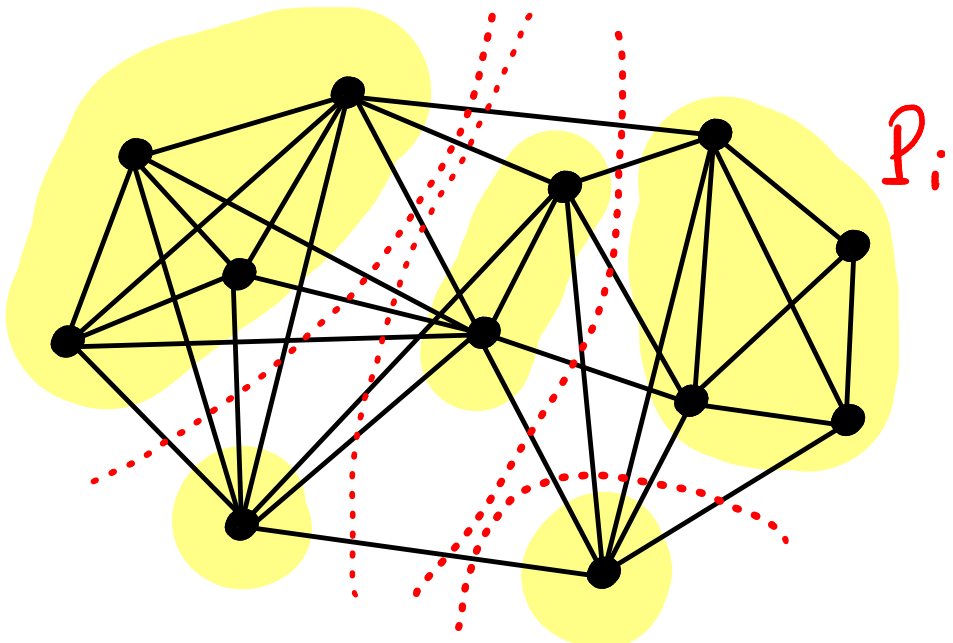


Kawarabayashi-Thorup partition

Theorem (Kawarabayashi-Thorup)

For simple graphs, there exists partition $V = P_1 \cup \cdots \cup P_k$ such that

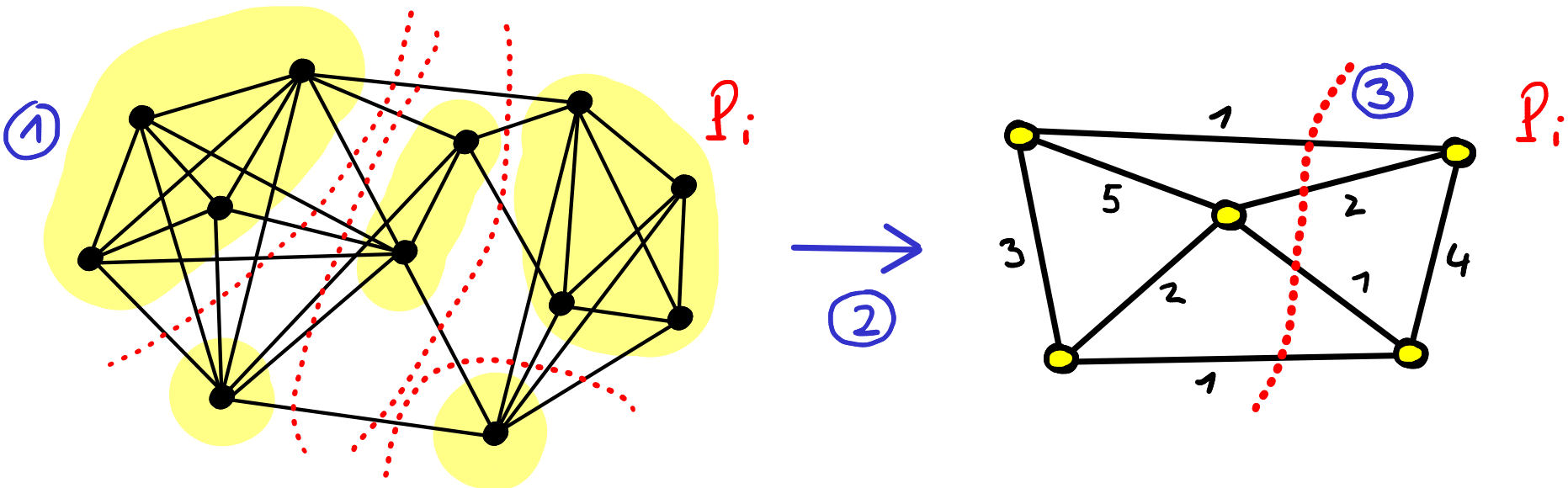
1. *there are $O(n)$ edges between partitions,*
2. *partition "respects" all (nontrivial) near-minimum cuts.*



Quantum query algorithm

Algorithm 1 Quantum query algorithm for MIN CUT

1:

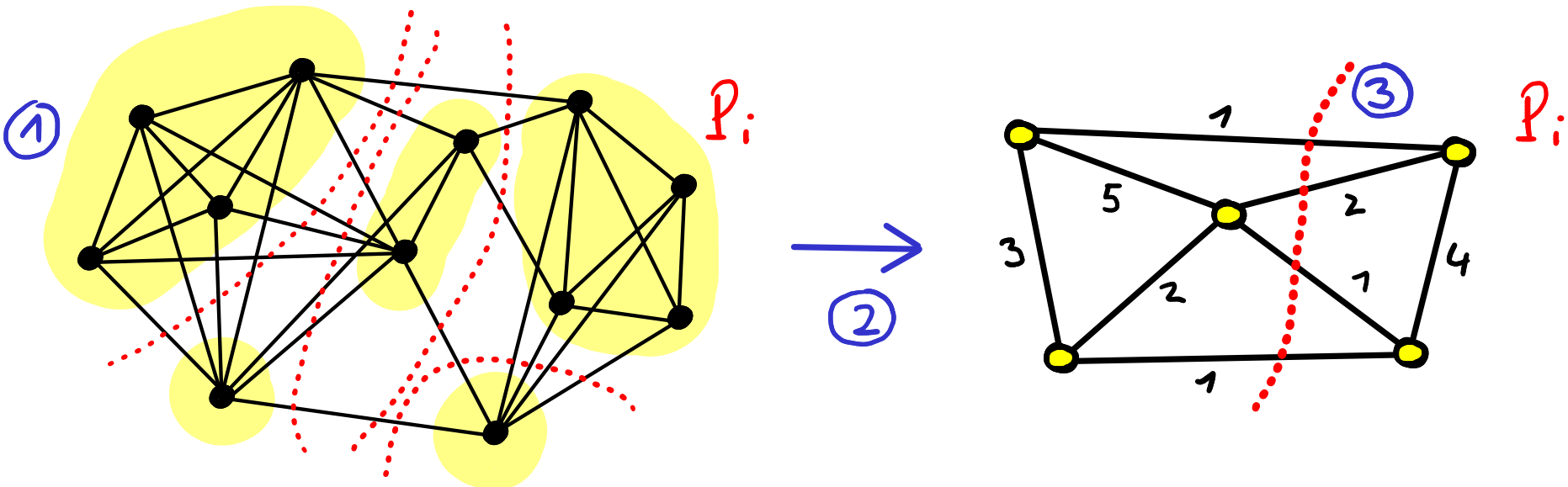


Quantum query algorithm

Algorithm 2 Quantum query algorithm for MIN CUT

1: construct KT partition $V = P_1 \cup \cdots \cup P_k$

▷ ?? queries



Quantum query algorithm

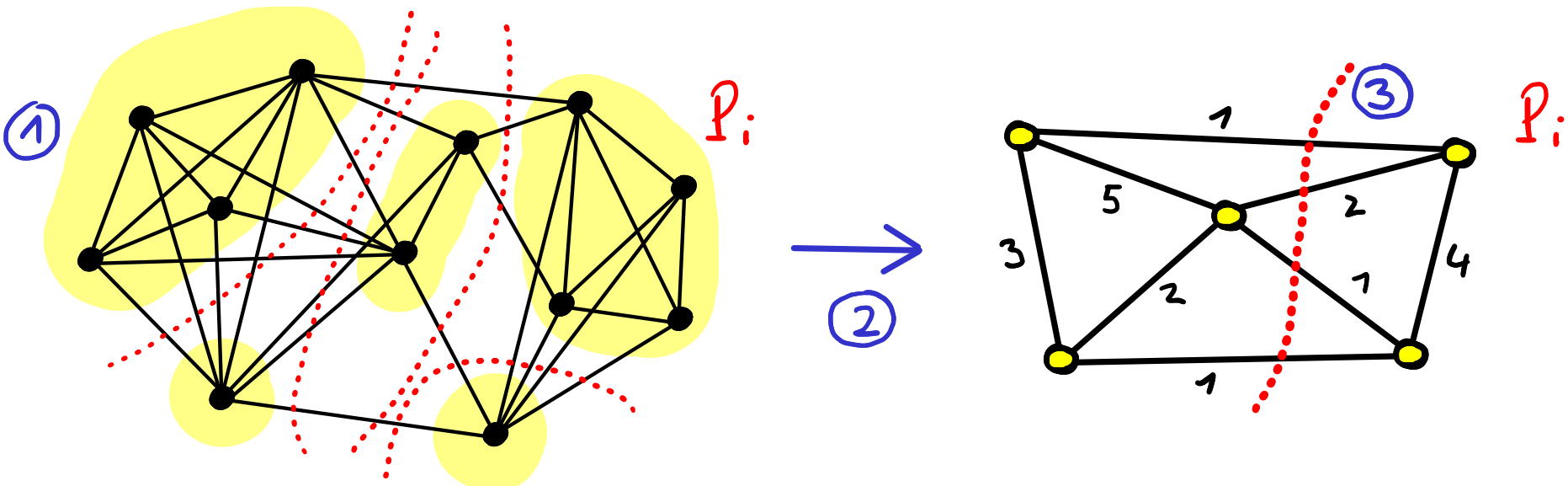
Algorithm 3 Quantum query algorithm for MIN CUT

1: construct KT partition $V = P_1 \cup \cdots \cup P_k$

▷ ?? queries

2: find $O(n)$ edges between partitions

▷ $O(n^{3/2})_M / O(\sqrt{mn})_L$ queries



Quantum query algorithm

Algorithm 4 Quantum query algorithm for MIN CUT

1: construct KT partition $V = P_1 \cup \cdots \cup P_k$

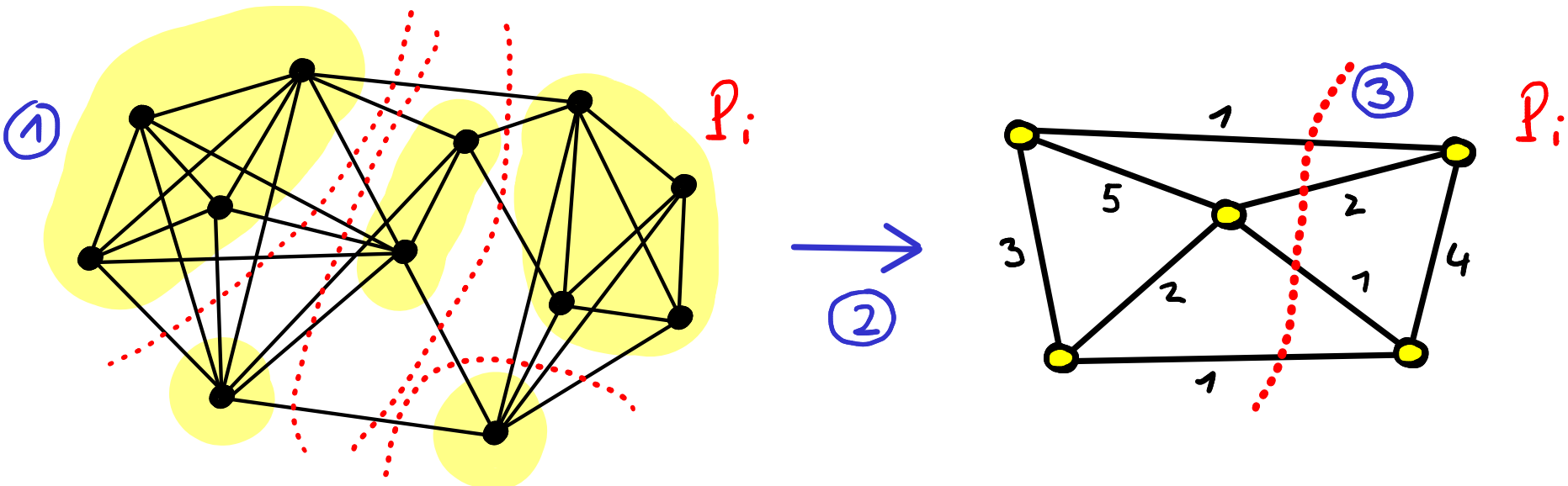
▷ ?? queries

2: find $O(n)$ edges between partitions

▷ $O(n^{3/2})_M / O(\sqrt{mn})_L$ queries

3: find minimum cut of contracted graph

▷ no queries



Quantum query algorithm

1: construct KT partition $V = P_1 \cup \cdots \cup P_k$

▷ ?? queries

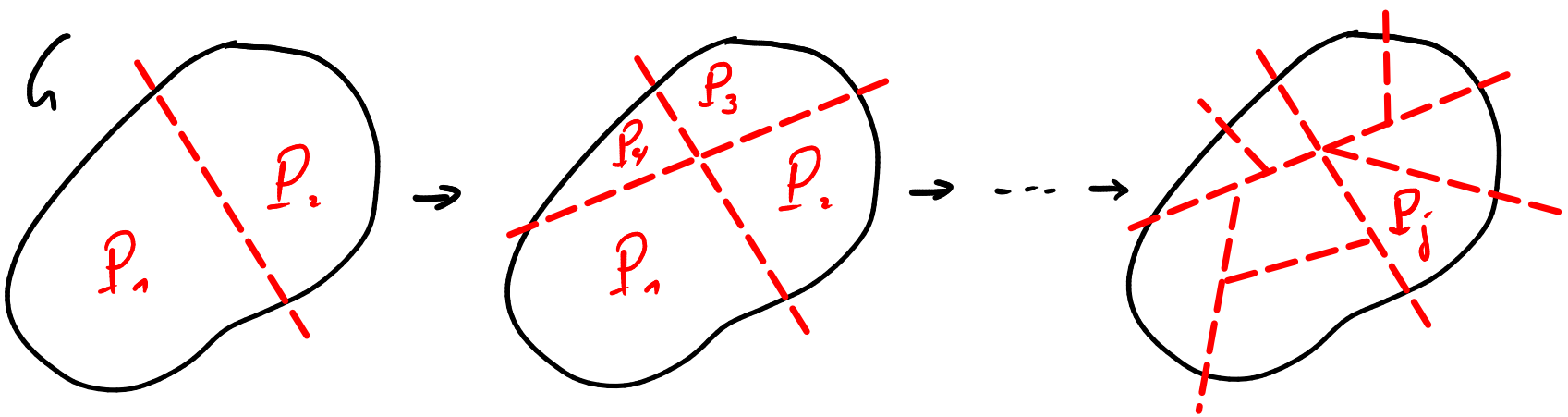
Quantum query algorithm

1: construct KT partition $V = P_1 \cup \cdots \cup P_k$

▷ ?? queries

---

greedy algorithm:
repeatedly refine partition by iterating over near-minimum cuts



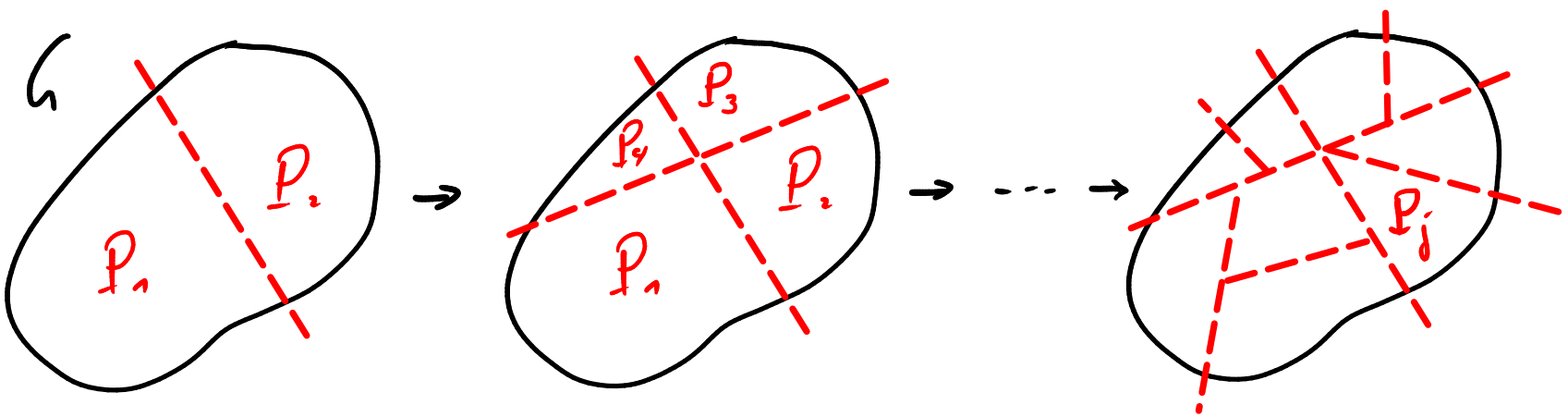
Quantum query algorithm

1: construct KT partition $V = P_1 \cup \cdots \cup P_k$

▷ ?? queries

---

greedy algorithm:
repeatedly refine partition by iterating over near-minimum cuts

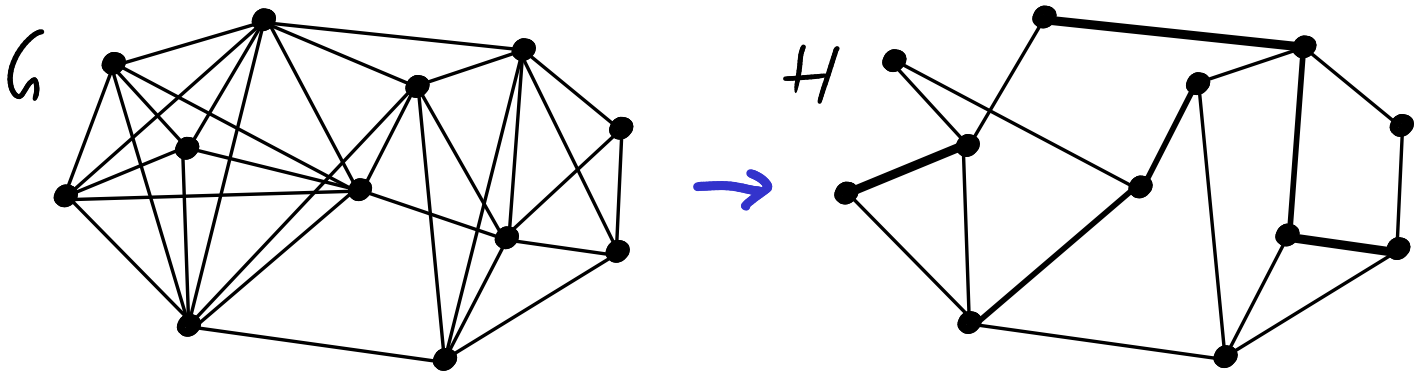


too costly!

Quantum query algorithm

observation:

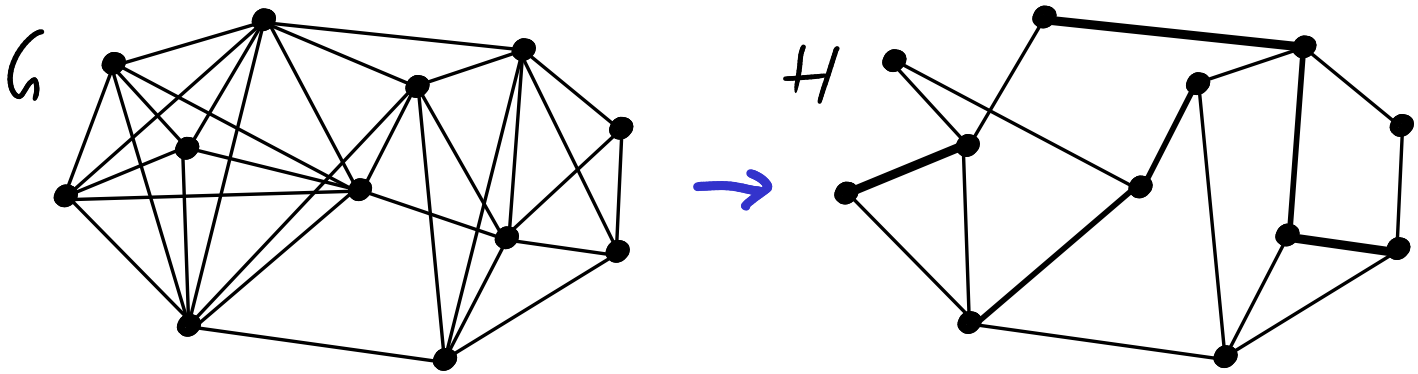
cut values in $G \approx$ cut values in sparsifier H



Quantum query algorithm

observation:

cut values in $G \approx$ cut values in sparsifier H



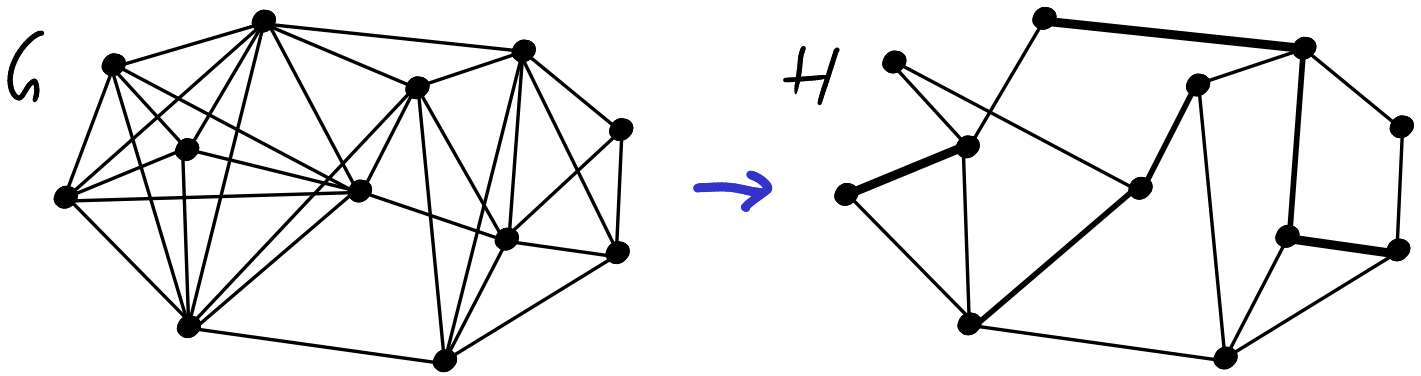
hence,

KT partition of $H \approx$ KT partition of G

Quantum query algorithm

observation:

cut values in $G \approx$ cut values in sparsifier H



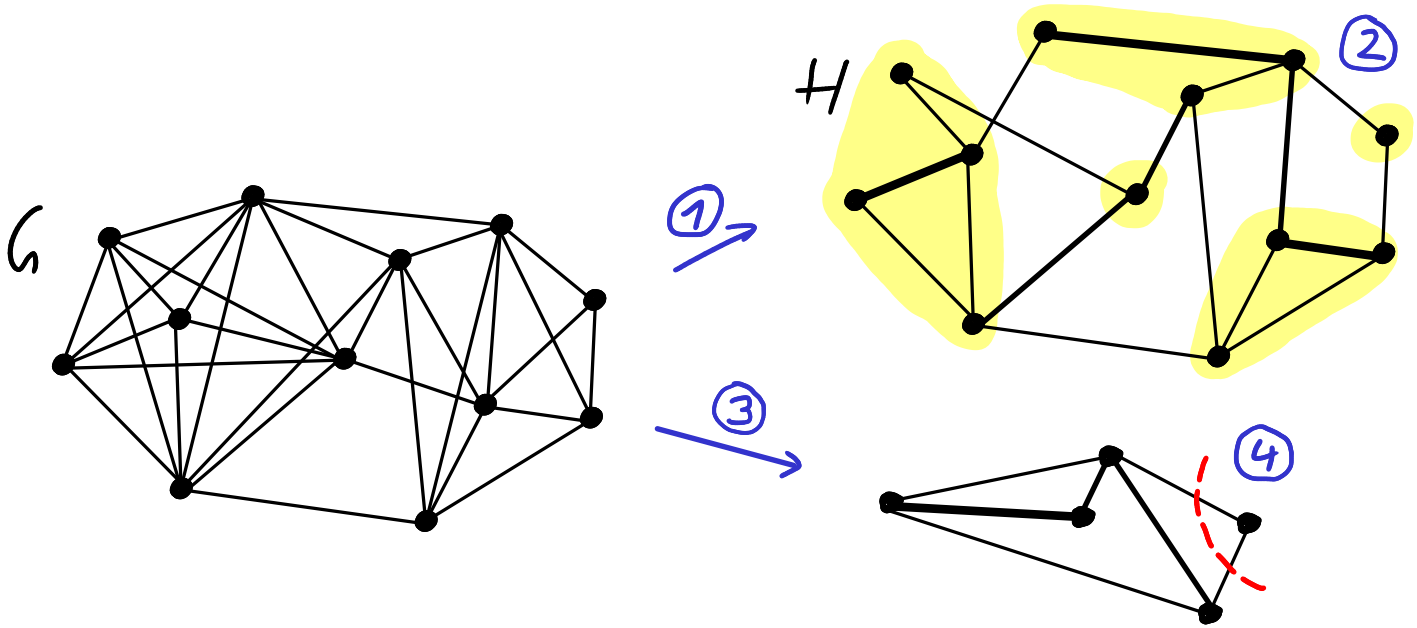
(query access) (explicit!)

hence,

KT partition of $H \approx$ KT partition of G

Quantum query algorithm for minimum cut

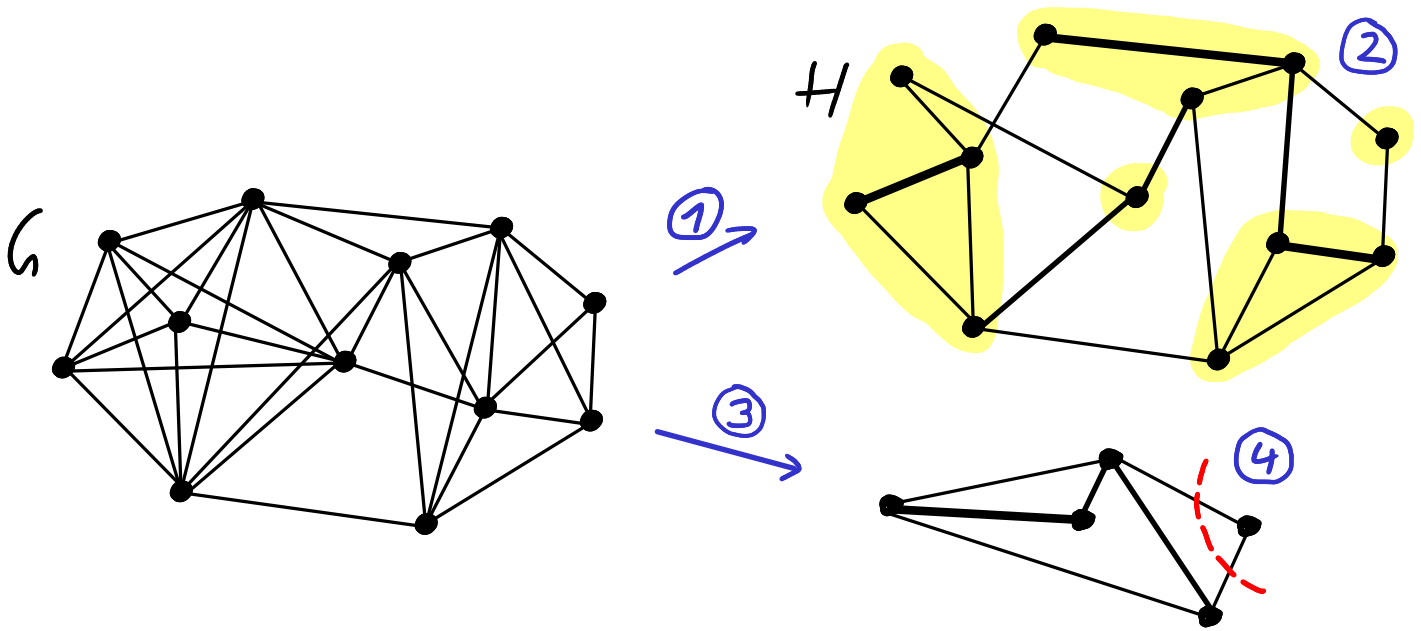
1:



Quantum query algorithm for minimum cut

1: Construct sparsifier H of G

▷ $\widetilde{O}(n^{3/2})_M / \widetilde{O}(\sqrt{mn})_L$ queries



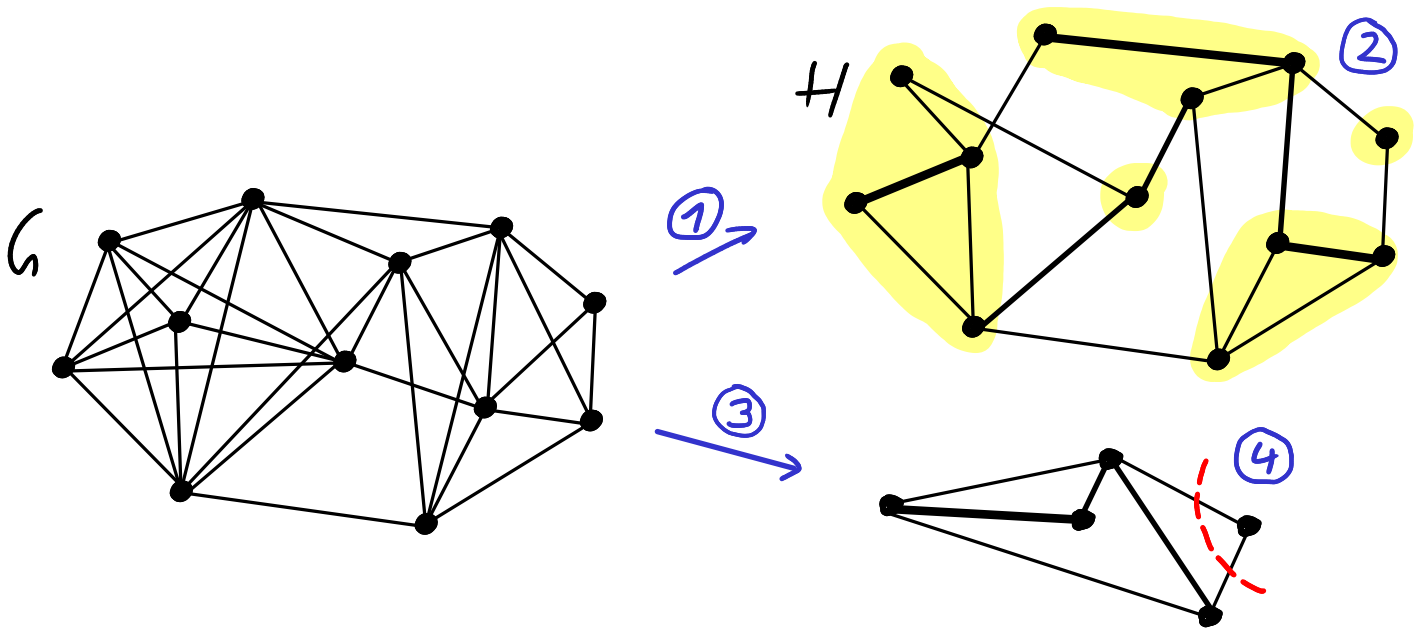
Quantum query algorithm for minimum cut

1: Construct sparsifier H of G

▷ $\widetilde{O}(n^{3/2})_M / \widetilde{O}(\sqrt{mn})_L$ queries

2: construct KT partition of H

▷ no queries



Quantum query algorithm for minimum cut

1: Construct sparsifier H of G

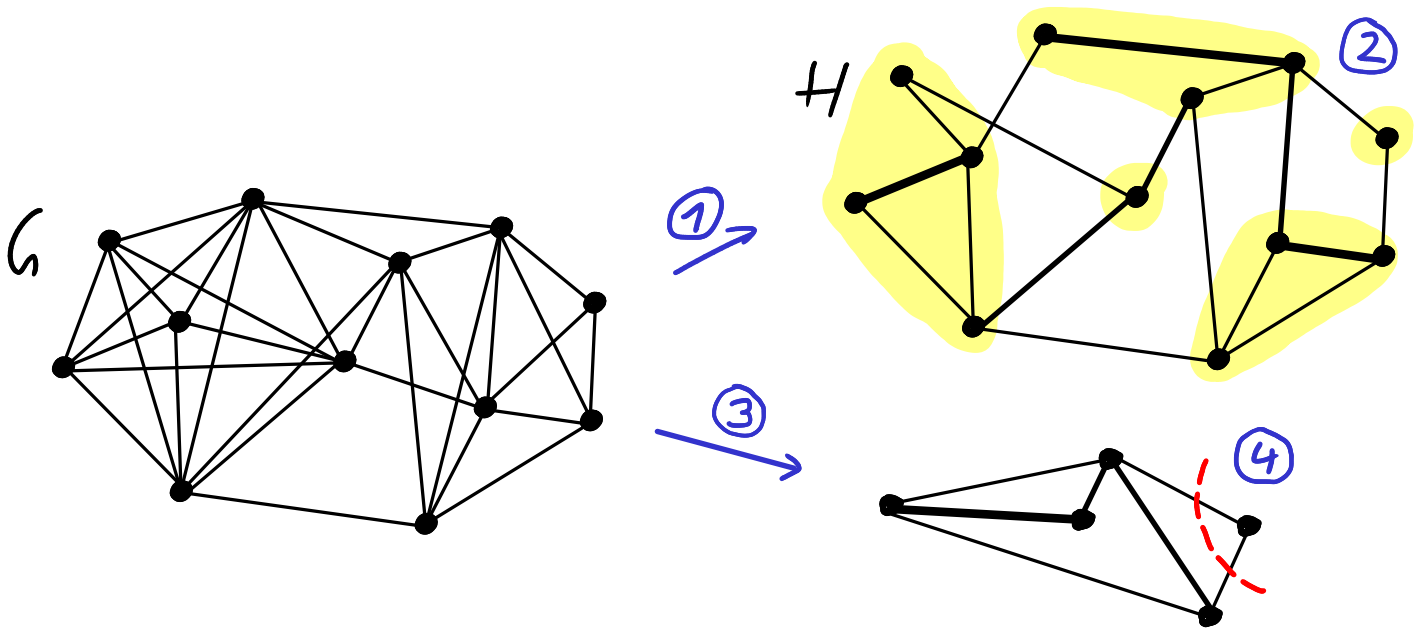
▷ $\widetilde{O}(n^{3/2})_M / \widetilde{O}(\sqrt{mn})_L$ queries

2: construct KT partition of H

▷ no queries

3: find $O(n)$ edges of G between partitions

▷ $O(n^{3/2})_M / \widetilde{O}(\sqrt{mn})_L$ queries



Quantum query algorithm for minimum cut

1: Construct sparsifier H of G

▷ $\widetilde{O}(n^{3/2})_M / \widetilde{O}(\sqrt{mn})_L$ queries

2: construct KT partition of H

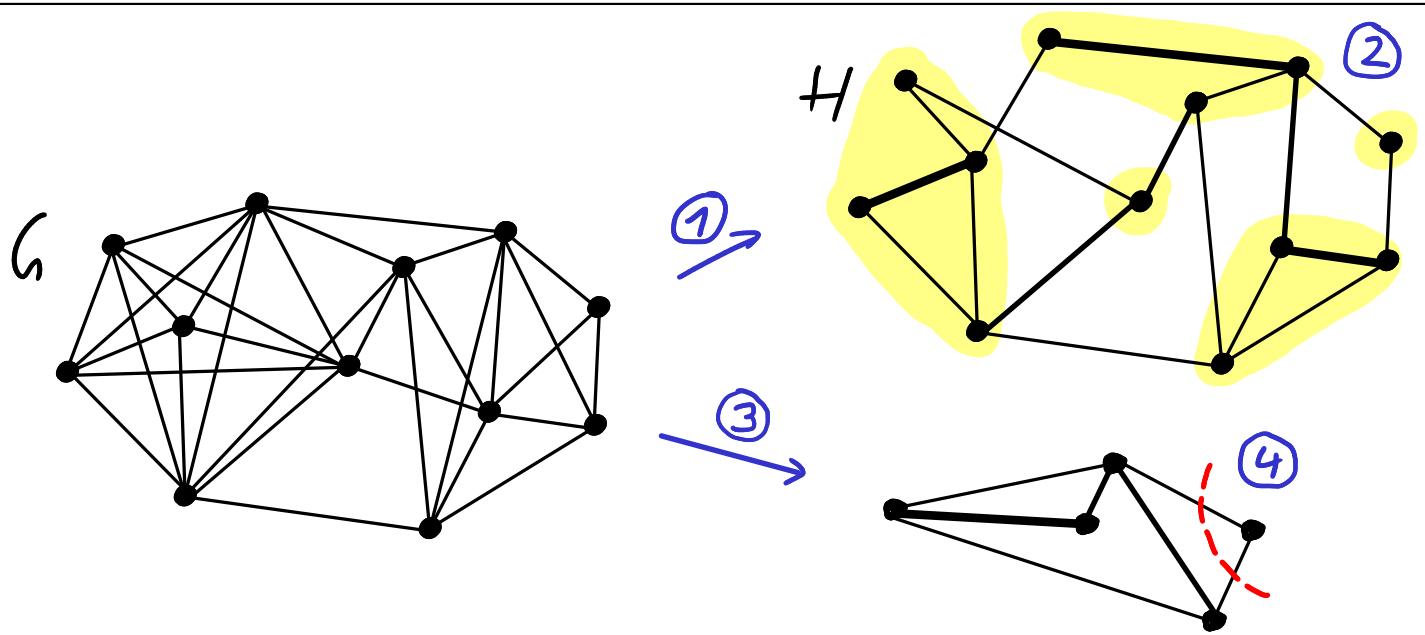
▷ no queries

3: find $O(n)$ edges of G between partitions

▷ $O(n^{3/2})_M / \widetilde{O}(\sqrt{mn})_L$ queries

4: solve minimum cut on contracted graph in time $\widetilde{O}(n)$

▷ no queries



Adjacency matrix model

Adjacency matrix model

algorithm for minimum cut with quantum query complexity $\widetilde{O}(n^{3/2})$

+

$\Omega(n^{3/2})$ lower bound on quantum query complexity of connectivity
(Dürr-Heiligman-Høyer-Mhalla '06)

Adjacency matrix model

algorithm for minimum cut with quantum query complexity $\widetilde{O}(n^{3/2})$

+

$\Omega(n^{3/2})$ lower bound on quantum query complexity of connectivity
(Dürr-Heiligman-Høyer-Mhalla '06)

=

Theorem

The quantum query complexity of minimum cut in simple graphs is

$$\widetilde{\Theta}(n^{3/2}) \text{ [M].}$$

Adjacency list model:

Adjacency list model:

algorithm for minimum cut with quantum query complexity $\widetilde{O}(\sqrt{mn})$

+

$\Omega(n)$ lower bound on quantum query complexity of connectivity
(Dürr-Heiligman-Høyer-Mhalla '06)

Adjacency list model:

algorithm for minimum cut with quantum query complexity $\widetilde{O}(\sqrt{mn})$

+

$\Omega(n)$ lower bound on quantum query complexity of connectivity
(Dürr-Heiligman-Høyer-Mhalla '06)

=

Theorem (adjacency list)

The quantum query complexity of minimum cut in simple graphs is

$$\widetilde{O}(\sqrt{mn})\ \textit{[L]} \quad \textit{and} \quad \Omega(n)\ \textit{[L]}.$$

Adjacency list model:

algorithm for minimum cut with quantum query complexity $\widetilde{O}(\sqrt{mn})$

+

$\Omega(n)$ lower bound on quantum query complexity of connectivity
(Dürr-Heiligman-Høyer-Mhalla '06)

=

Theorem (adjacency list)

The quantum query complexity of minimum cut in simple graphs is

$$\widetilde{O}(\sqrt{mn})\ \textit{[L]} \quad \textit{and} \quad \Omega(n)\ \textit{[L]}.$$

! open question: $\widetilde{O}(n)$ quantum algorithm ?

Time efficient quantum algorithm for minimum cut

based on [A - Lee - Gawrychowski]

Quantum algorithm for minimum cut (adj.list)

1: Construct sparsifier H of G

▷ time $\widetilde{O}(n^{3/2})_M / \widetilde{O}(\sqrt{mn})_L$ [A-de Wolf '20]

2: construct KT partition of H

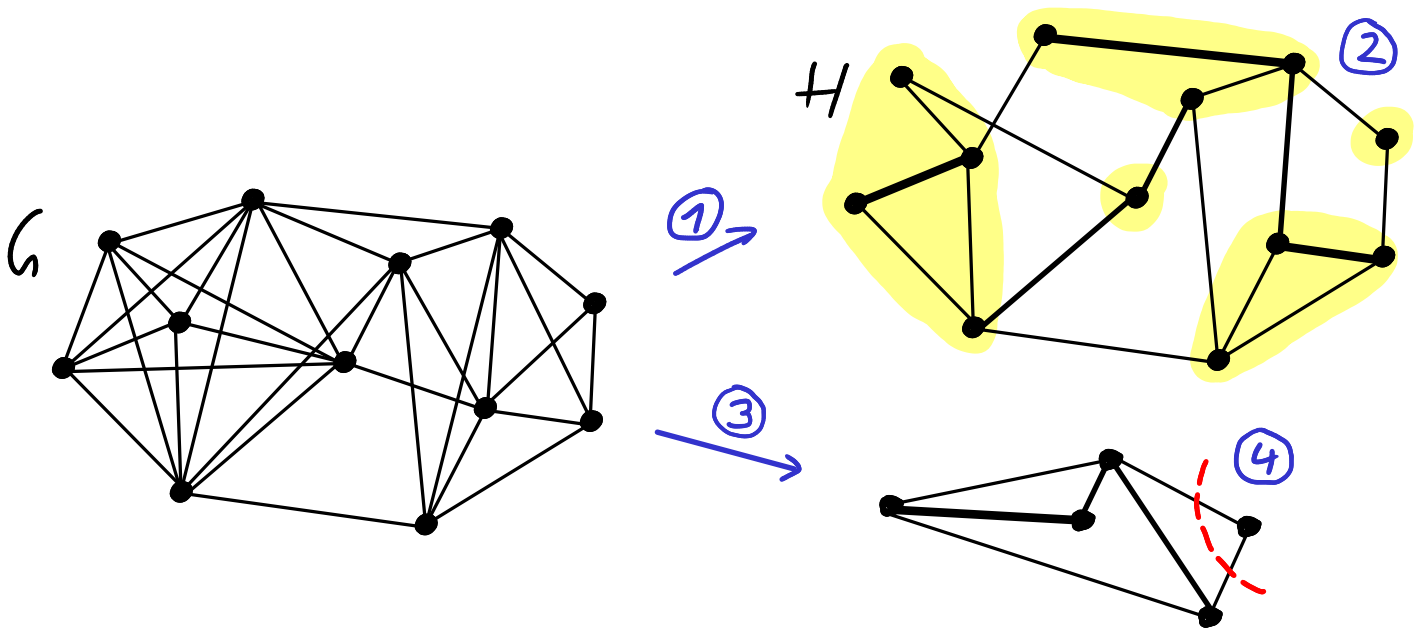
▷ time ???

3: find $O(n)$ edges of G between partitions

▷ time $O(n^{3/2})_M / O(\sqrt{mn})_L$

4: solve MIN CUT on contracted graph in time $\widetilde{O}(n)$

▷ time $\widetilde{O}(n)$ [Karger '00]



Quantum algorithm for minimum cut

1: construct KT partition of H

▷ time ???

Quantum algorithm for minimum cut

1: construct KT partition of H

▷ time ???

•

Kawarabayashi-Thorup (STOC '15):
classical algorithm for simple graphs in time $\widetilde{O}(m)$

Quantum algorithm for minimum cut

1: construct KT partition of H

▷ time ???

•

Kawarabayashi-Thorup (STOC '15):
classical algorithm for simple graphs in time $\widetilde{O}(m)$

•

A-Lee (CCC '21):
quantum algorithm for weighted graphs in time $\widetilde{O}(n^{3/2})$ [M,L]

Quantum algorithm for minimum cut

1: construct KT partition of H

▷ time ???

•

Kawarabayashi-Thorup (STOC '15):
classical algorithm for simple graphs in time $\widetilde{O}(m)$

•

A-Lee (CCC '21):
quantum algorithm for weighted graphs in time $\widetilde{O}(n^{3/2})$ [M,L]

•

A-Gawrychowski-Lee (forthcoming):
classical algorithm for weighted graphs in time $\widetilde{O}(m)$
quantum algorithm for weighted graphs in time $\widetilde{O}(\sqrt{mn})$ [L]

Classical algorithm for KT partition

key ideas:

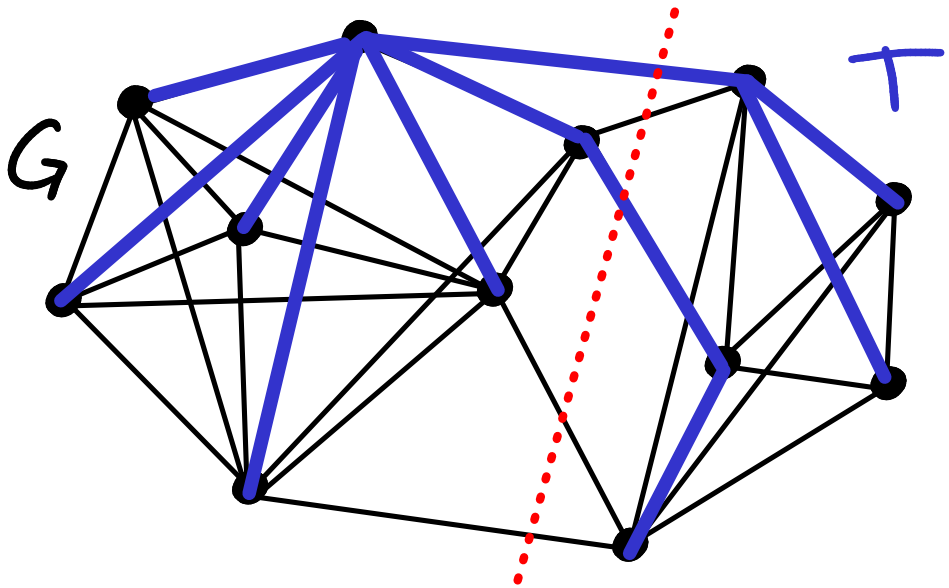
1. “tree-respecting cuts”

2. generating set for KT partition

Tree-respecting cuts

introduced by Karger (J. ACM '00)

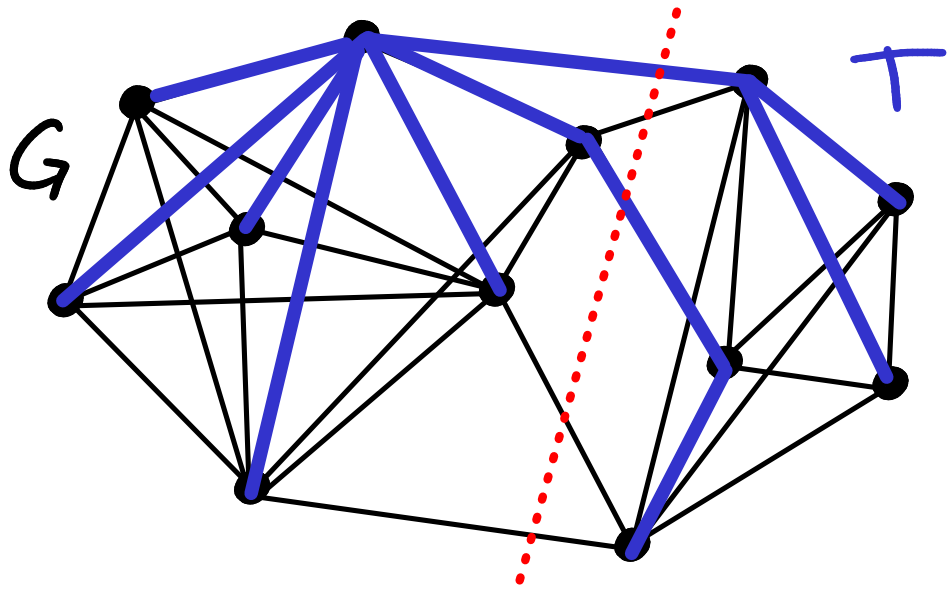
graph G, spanning tree T



Tree-respecting cuts

introduced by Karger (J. ACM '00)

graph G, spanning tree T

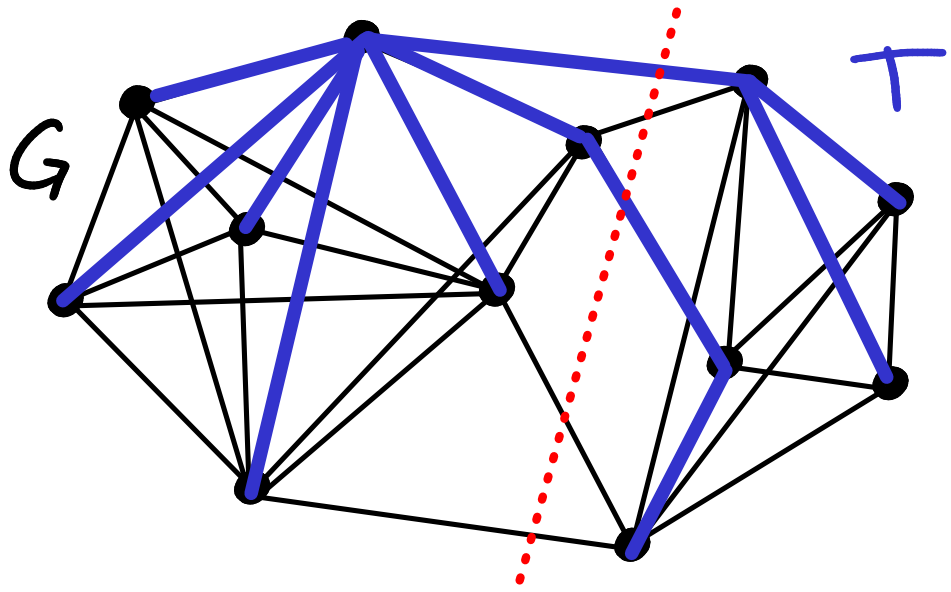


cut "2-respects" T if cuts at most 2 edges of T

Tree-respecting cuts

introduced by Karger (J. ACM '00)

graph G, spanning tree T



cut "2-respects" T if cuts at most 2 edges of T

! at most $\binom{n-1}{2}$ 2-respecting cuts

Tree-respecting cuts

Lemma (Karger)

Any near-minimum cut 2-respects "random" spanning tree T with large probability.

Tree-respecting cuts

Lemma (Karger)

Any near-minimum cut 2-respects "random" spanning tree T with large probability.

corollary: at most $\widetilde{O}(n^2)$ near-minimum cuts

Tree-respecting cuts

Lemma (Karger)

Any near-minimum cut 2-respects "random" spanning tree T with large probability.

corollary: at most $\widetilde{O}(n^2)$ near-minimum cuts

$\downarrow$

"better" greedy algorithm for KT partition:
iterate over 2-respecting cuts

Tree-respecting cuts

Lemma (Karger)

Any near-minimum cut 2-respects "random" spanning tree T with large probability.

corollary: at most $\widetilde{O}(n^2)$ near-minimum cuts

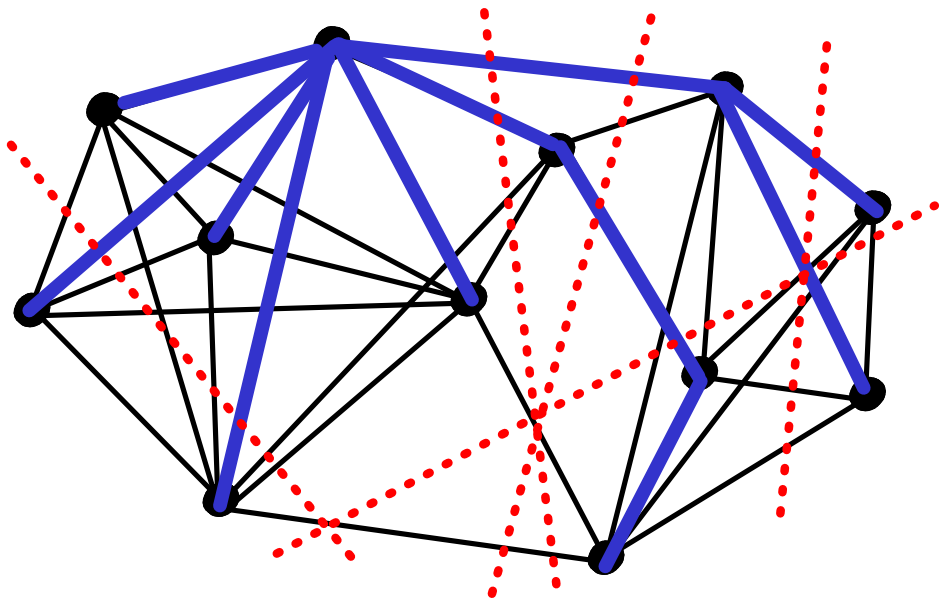
$\downarrow$

"better" greedy algorithm for KT partition:
iterate over 2-respecting cuts

! still too slow

Generating set

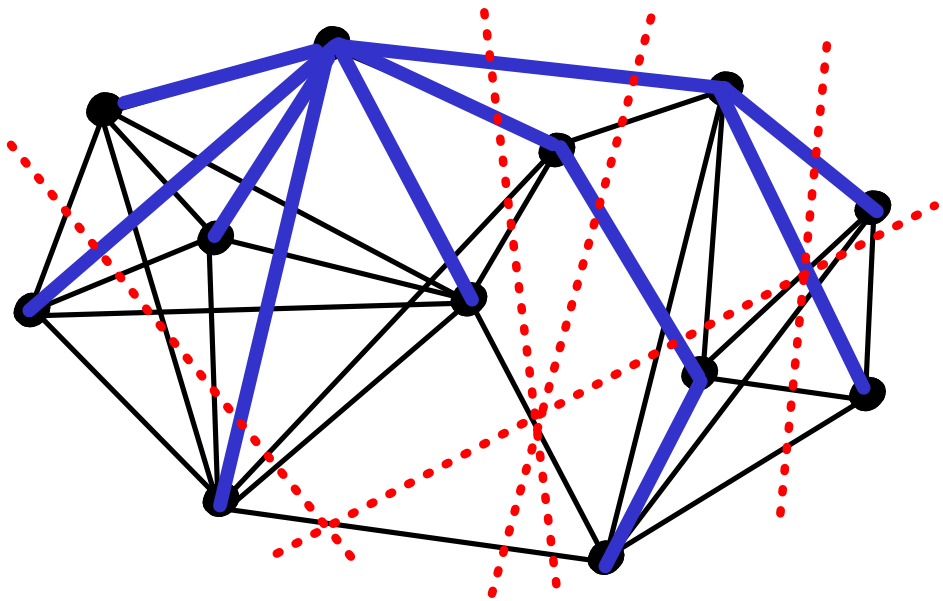
Lemma (A-Lee ’20, A-Gawrychowski-Lee ’21)



Generating set

Lemma (A-Lee '20, A-Gawrychowski-Lee '21)

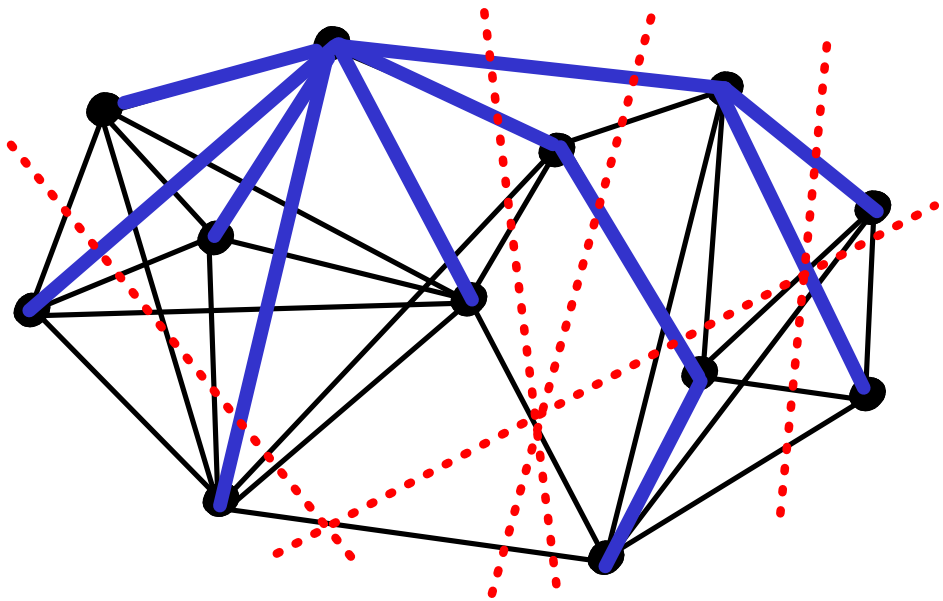
- *Characterize "generating" set of $O(n)$ 2-respecting cuts that induces KT partition*



Generating set

Lemma (A-Lee '20, A-Gawrychowski-Lee '21)

- *Characterize "generating" set of $O(n)$ 2-respecting cuts that induces KT partition*
- *Find generating set in time $\widetilde{O}(m)$ (classically) or $\widetilde{O}(\sqrt{mn})$ (quantum)*



Generating set

Theorem

Can construct the KT partition of a weighted graph in time $\widetilde{O}(m)$ (classically) or $\widetilde{O}(\sqrt{mn})$ (quantumly).

Generating set

Theorem

Can construct the KT partition of a weighted graph in time $\widetilde{O}(m)$ (classically) or $\widetilde{O}(\sqrt{mn})$ (quantumly).

Corollary

There is a quantum algorithm for finding a minimum cut in a simple graph with **time complexity** $\widetilde{O}(n^{3/2})_M / \widetilde{O}(\sqrt{mn})_L$.

Summary

Quantum time/query complexity of minimum cut:

Summary

Quantum time/query complexity of minimum cut:

weighted graphs, exact: no quantum speedup

Summary

Quantum time/query complexity of minimum cut:

weighted graphs, exact: no quantum speedup

weighted graphs, approximate: $\widetilde{O}(n^{3/2}/\epsilon)_M$ / $\widetilde{O}(\sqrt{mn}/\epsilon)_L$

Summary

Quantum time/query complexity of minimum cut:

weighted graphs, exact: no quantum speedup

weighted graphs, approximate: $\widetilde{O}(n^{3/2}/\epsilon)_M$ / $\widetilde{O}(\sqrt{mn}/\epsilon)_L$

simple graphs, exact: $\widetilde{\Theta}(n^{3/2})_M$ / $\widetilde{O}(\sqrt{mn})_L$

Summary

Quantum time/query complexity of minimum cut:

weighted graphs, exact: no quantum speedup

weighted graphs, approximate: $\widetilde{O}(n^{3/2}/\epsilon)_M$ / $\widetilde{O}(\sqrt{mn}/\epsilon)_L$

simple graphs, exact: $\widetilde{\Theta}(n^{3/2})_M$ / $\widetilde{O}(\sqrt{mn})_L$

Key tools:

graph sparsification and Kawarabayashi-Thorup partition

Summary

Quantum time/query complexity of minimum cut:

weighted graphs, exact: no quantum speedup

weighted graphs, approximate: $\widetilde{O}(n^{3/2}/\epsilon)_M$ / $\widetilde{O}(\sqrt{mn}/\epsilon)_L$

simple graphs, exact: $\widetilde{\Theta}(n^{3/2})_M$ / $\widetilde{O}(\sqrt{mn})_L$

Key tools:

graph sparsification and Kawarabayashi-Thorup partition

Open questions:

$\widetilde{O}(n)_L$ quantum algorithms for
minimum cut and sparsification in simple graphs?

$\Omega(\sqrt{mn}/\epsilon)_L$ lower bound for approximate minimum cut?