Parameterized Quantum Query Algorithms for Graph Problems

Tatsuya Terao[1], Ryuhei Mori[2]

1. Kyoto University
2. Nagoya University

ESA 2024 @Egham Sep 4, 2024

Parameterized Quantum Query Algorithms for Graph Problems

vertex cover and matching

Tatsuya Terao[1], Ryuhei Mori[2]

1. Kyoto University
2. Nagoya University

ESA 2024 @Egham Sep 4, 2024

kernelization and augmenting paths

Parameterized Quantum Query Algorithms for Graph Problems

vertex cover and matching

Tatsuya Terao[1], Ryuhei Mori[2]

1. Kyoto University
2. Nagoya University

ESA 2024 @Egham Sep 4, 2024

Query Complexity

Given f as an oracle !

Oracle $\mathcal{O}_f$

$f: \{1, \dots, N\} \to \{0, 1\}$

Query Complexity

Algorithm

ask $f(i)$

output $f(i)$

Oracle $\mathcal{O}_f$

$f: \{1, \ldots, N\} \rightarrow \{0, 1\}$

☞ Query Complexity = # of queries to oracle

Quantum Query Complexity

Quantum Algo

$|\boldsymbol{i}, b\rangle$

$|i, b \oplus \boldsymbol{f}(\boldsymbol{i})\rangle$

Quantum Oracle $\mathcal{O}_f$

$\boldsymbol{f}: \{\mathbf{1}, \ldots, \boldsymbol{N}\} \rightarrow \{\mathbf{0}, \mathbf{1}\}$

☞ Query Complexity = # of $\mathcal{O}_f$

Quantum Query Complexity

Quantum Algo

$\sum_i \alpha_i \, |\boldsymbol{i}, b\rangle$

$\sum_i \alpha_i \, |i, b \oplus \boldsymbol{f}(\boldsymbol{i})\rangle$

Quantum Oracle $\mathcal{O}_f$

$\boldsymbol{f}: \{\mathbf{1}, \dots, \boldsymbol{N}\} \to \{\mathbf{0}, \mathbf{1}\}$

☞ Query Complexity = # of $\mathcal{O}_f$

Grover's Algorithm

Input : Oracle access to $\boldsymbol{f}\colon \{\mathbf{1}, \ldots, \boldsymbol{N}\} \rightarrow \{\mathbf{0}, \mathbf{1}\}$
Output : $i \in \{1, \ldots, N\}$ s.t. $f(i) = 1$

Grover's Algorithm

Input : Oracle access to $\boldsymbol{f}: \{\mathbf{1}, \dots, \boldsymbol{N}\} \rightarrow \{\mathbf{0}, \mathbf{1}\}$
Output : $i \in \{1, \dots, N\}$ s.t. $f(i) = 1$

Classical

$\boldsymbol{\Theta}(\boldsymbol{N})$ queries
with error prob. at most 1/3

Quantum

Grover's Algorithm

Input : Oracle access to $f: \{1, \dots, N\} \to \{0, 1\}$
Output : $i \in \{1, \dots, N\}$ s.t. $f(i) = 1$

Classical

$\Theta(N)$ queries
with error prob. at most 1/3

Quantum

$O(\sqrt{N})$ queries with error prob. at most 1/3 [Grover '96]

Lower Bound: $\Omega(\sqrt{N})$
[Bennett-Bernstein-Brassard-Vazirani '97]

Grover's Algorithm

Input : Oracle access to $\boldsymbol{f}: \{\mathbf{1}, \dots, \boldsymbol{N}\} \rightarrow \{\mathbf{0}, \mathbf{1}\}$
Output : $i \in \{1, \dots, N\}$ s.t. $f(i) = 1$

Classical

$\boldsymbol{\Theta}(\boldsymbol{N})$ queries
with error prob. at most 1/3

Quantum

Classical-Quantum Separation !

$\boldsymbol{O}(\sqrt{\boldsymbol{N}})$ queries with error prob.
at most 1/3 [Grover '96]

Limitation of Quantum Algo

Lower Bound: $\boldsymbol{\Omega}(\sqrt{\boldsymbol{N}})$
[Bennett-Bernstein-Brassard-Vazirani '97]

Quantum Query Complexity for Graph Problems

Adjacency Matrix Model

Quantum oracle access to $E_M\colon \{1,\ldots,n\}\times\{1,\ldots,n\} \to \{0,1\}$

$$\boldsymbol{E_M(u,v) = 1 \Leftrightarrow (u,v) \in E(G)}$$

$n = \#$ of vertices

Previous Works on Query Complexity

Even through classical algorithms require $\boldsymbol{\Theta(n^2)}$ queries, …

- k-clique : $\boldsymbol{\tilde{O}(n^{2-2/k})}$ [Magniez-Santha-Szegedy '05]
- Connectivity : $\boldsymbol{\Theta(n^{3/2})}$ [Dürr-Heiligman-Høyer-Mhalla '06]
- Planarity : $\boldsymbol{\Theta(n^{3/2})}$ [Ambainis et al. '08]
- Maximum matching : $\boldsymbol{O(n^{7/4})}$ [Kimmel-Witter '21], $\Omega(n^{3/2})$ [Zhang '04]
- Minimum cut : $\boldsymbol{\Theta(n^{3/2})}$ [Apers-Lee '21]

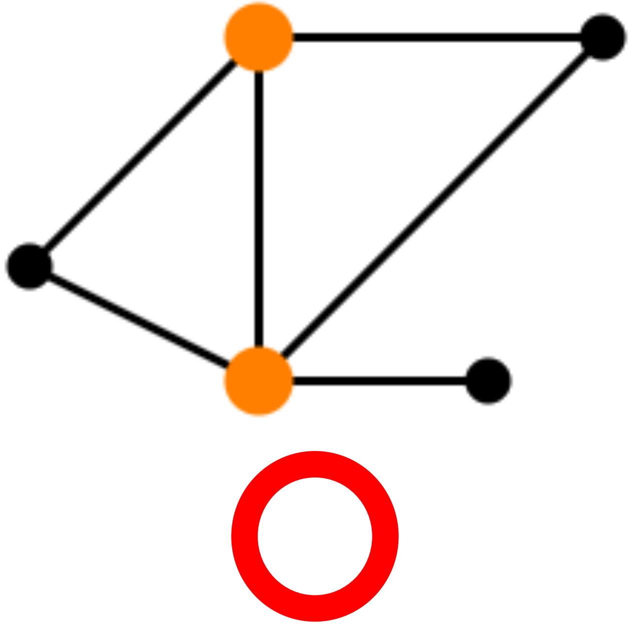
n = # of vertices

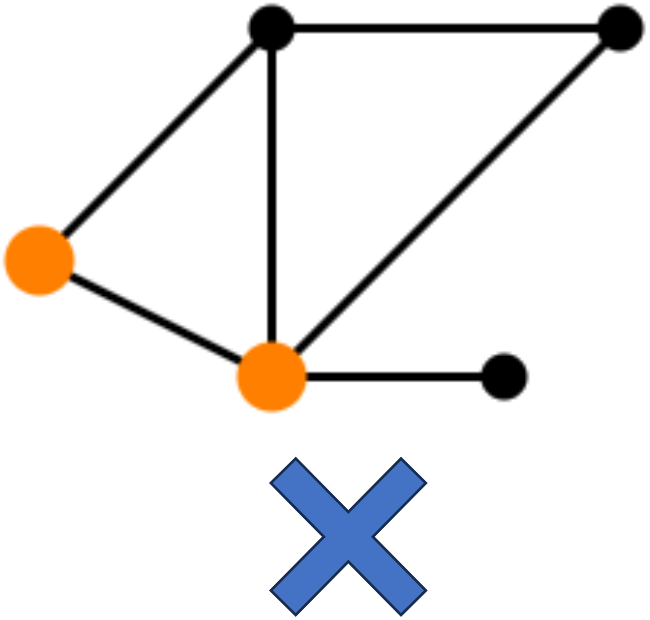
k-vertex cover problem

Input : an undirected graph G and an interger k
Find : a **vertex cover** $S \subseteq V$ of size **at most** k

every edge of G has at least one endpoint in S

e.g.)





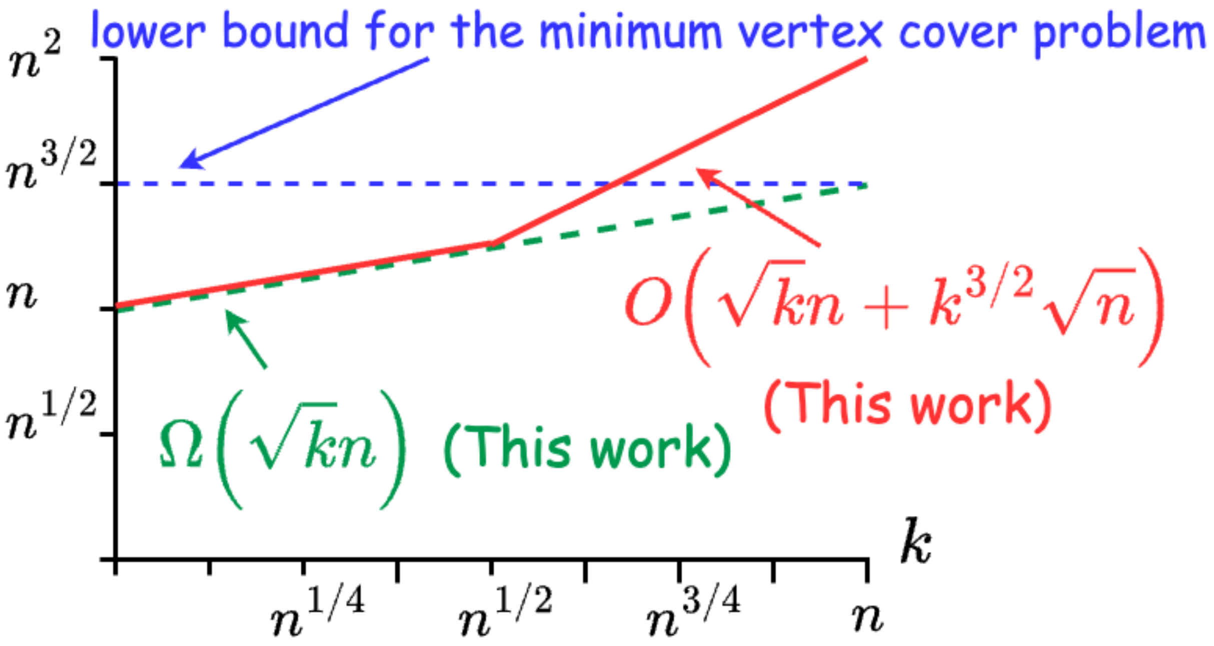
Our Contribution 1. Parameterized Quantum Query Complexity for Vertex Cover

Thm.

Quantum Query Complexity to find a vertex cover of size at most k

Upper Bound : $O(\sqrt{kn} + k^{3/2}\sqrt{n})$

Lower Bound : $\Omega(\sqrt{kn})$ (when $k \leq (1-\epsilon)n$)

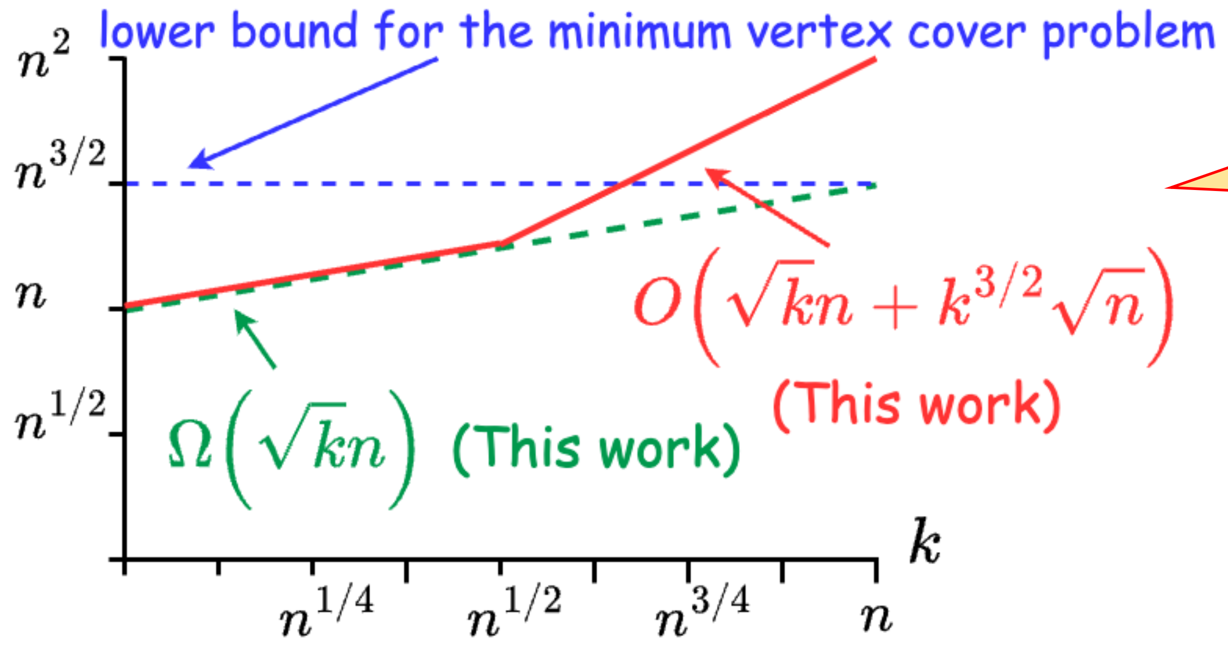


Our Contribution 1. Parameterized Quantum Query Complexity for Vertex Cover

Thm.
Quantum Query Complexity to find a vertex cover of size at most k

Upper Bound : $O(\sqrt{kn} + k^{3/2}\sqrt{n})$

Lower Bound : $\Omega(\sqrt{kn})$ (when $k \le (1-\epsilon)n$)



Optimal complexity
$\Theta(\sqrt{kn})$ when $k = O(\sqrt{n})$

Our Contribution 1. Parameterized Quantum Query Complexity for Vertex Cover

Thm.

Quantum Query Complexity to find a vertex cover of size at most k

Upper Bound : $O(\sqrt{k}n + k^{3/2}\sqrt{n})$

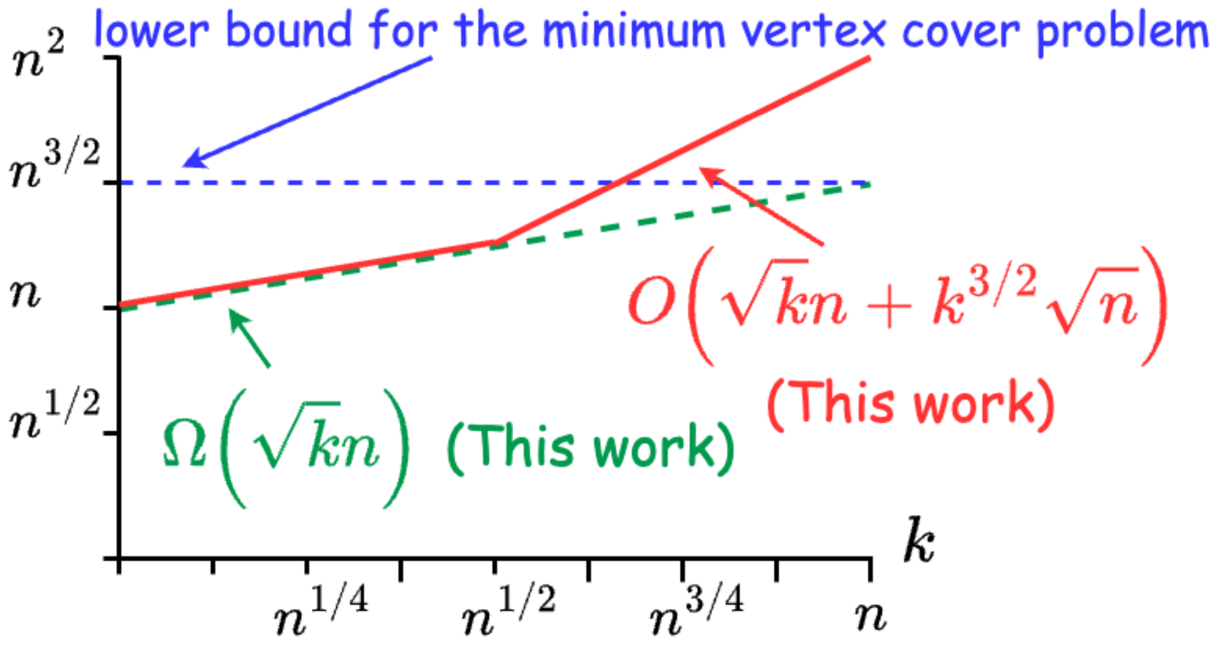
Lower Bound : $\Omega(\sqrt{k}n)$ (when $k \le (1-\epsilon)n$)

Significance

- UB $O(n^2)$, LB $\Omega(n^{3/2})$ [Zhang '04] were only known for minimum vertex cover
- Consider Parameterized ver.

Technique

- Quantum Query Kernelization



Kernelization

Input : instance (G, k)
Output : another equivalent small instance (G', k')

- (G, k) is a Yes instance $\Leftrightarrow$ (G', k') is a Yes instance
- $E(G') \leq f(k)$
- $k' \leq g(k)$

Buss-Goldsmith's Kernelization for k-vertex cover

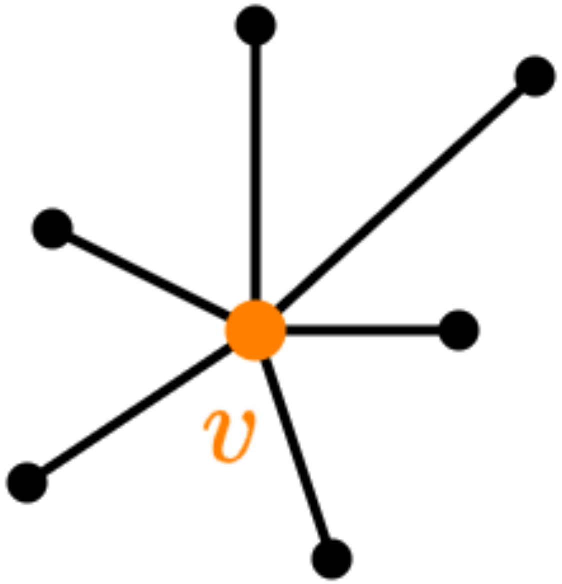
Rule 1. If G has an isolated vertex v, then $\boldsymbol{(G, k) \to (G - v, k)}$

Buss-Goldsmith's Kernelization for k-vertex cover

Rule 1. If G has an isolated vertex v, then $(G, k) \rightarrow (G - v, k)$

Rule 2. If G has a vertex v of **degree at least $k + 1$**, then

If v is not in a vertex cover, then it must contain all neighbors of v.

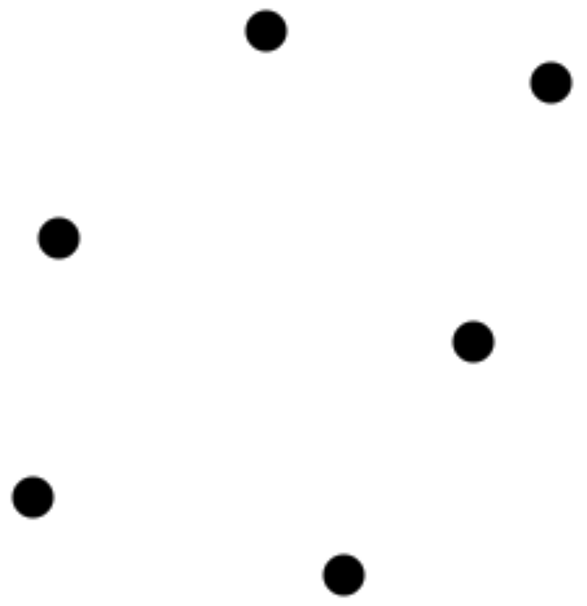


Buss-Goldsmith's Kernelization for k-vertex cover

Rule 1. If G has an isolated vertex v, then $(G, k) \to (G - v, k)$

Rule 2. If G has a vertex v of degree at least $k + 1$, then
$\boldsymbol{(G, k) \to (G - v, k - 1)}$

v must be in any vertex cover of size at most k.



Buss-Goldsmith's Kernelization for k-vertex cover

Rule 1. If G has an isolated vertex v, then $(G, k) \rightarrow (G - v, k)$

Rule 2. If G has a vertex v of degree at least $k + 1$, then
$(G, k) \rightarrow (G - v, k - 1)$

v must be in any vertex cover of size at most k.

Fact: After Applying Rules 1 and 2, if $|E(G)| > k^2$, then (G, k) is a No instance

New Approach: Quantum Query Kernelization

Input : Oracle access to (G, k)
Output : another equivalent instance (G', k') as a bit string

- (G, k) is a Yes instance $\Leftrightarrow$ (G', k') is a Yes instance

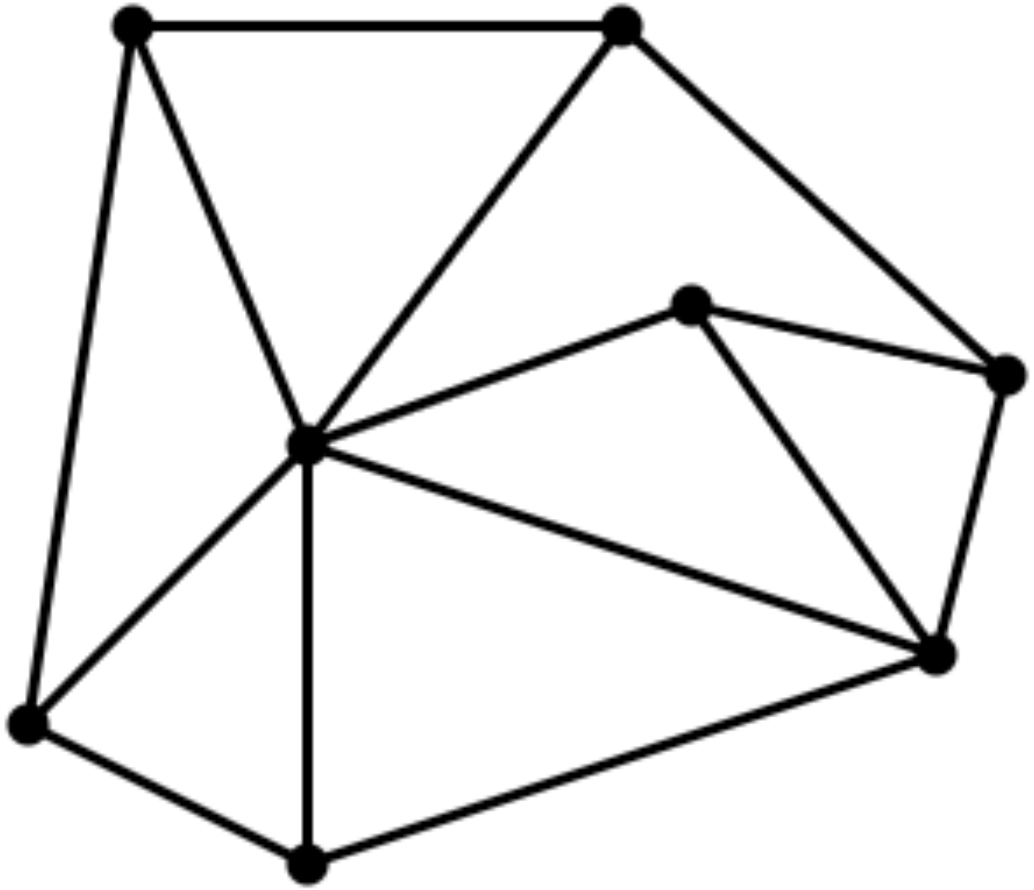
New Approach: Quantum Query Kernelization

Input : Oracle access to (G, k)
Output : another equivalent instance (G', k') as a bit string

- (G, k) is a Yes instance $\Leftrightarrow$ (G', k') is a Yes instance

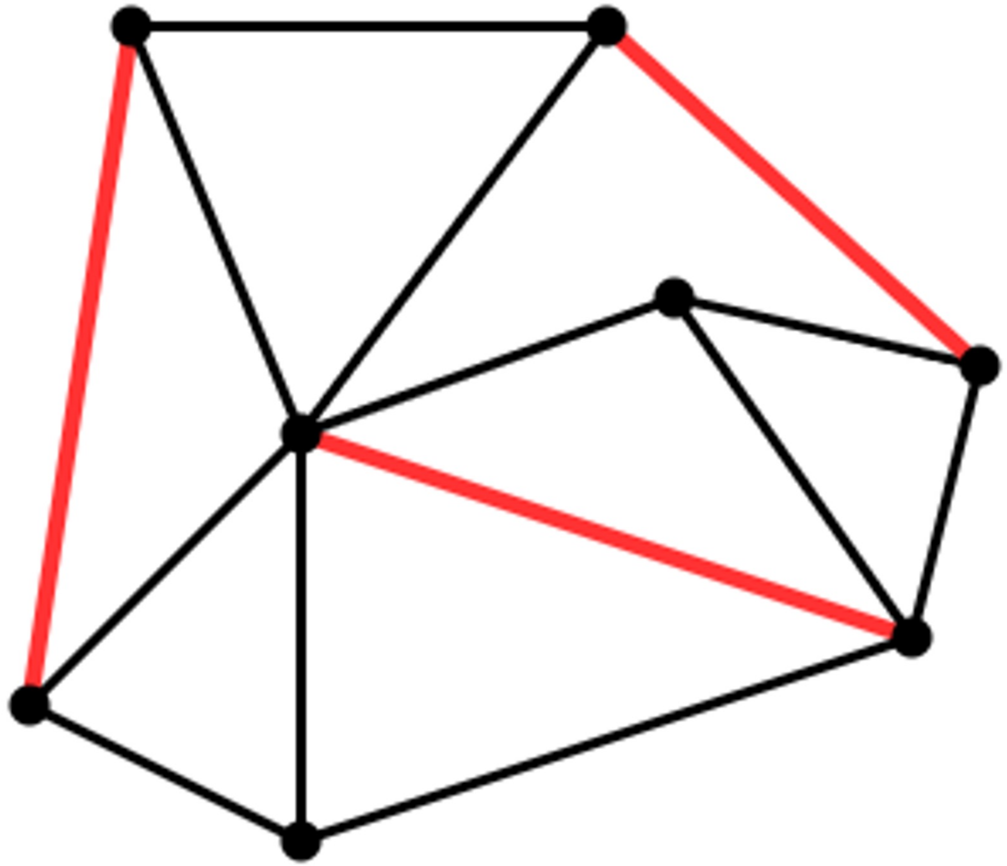
☞ After Applying quantum query kernelization,
just apply classical algorithm for (G', k').

Classical Kernelization Suitable for Quantum Algo



Classical Kernelization Suitable for Quantum Algo

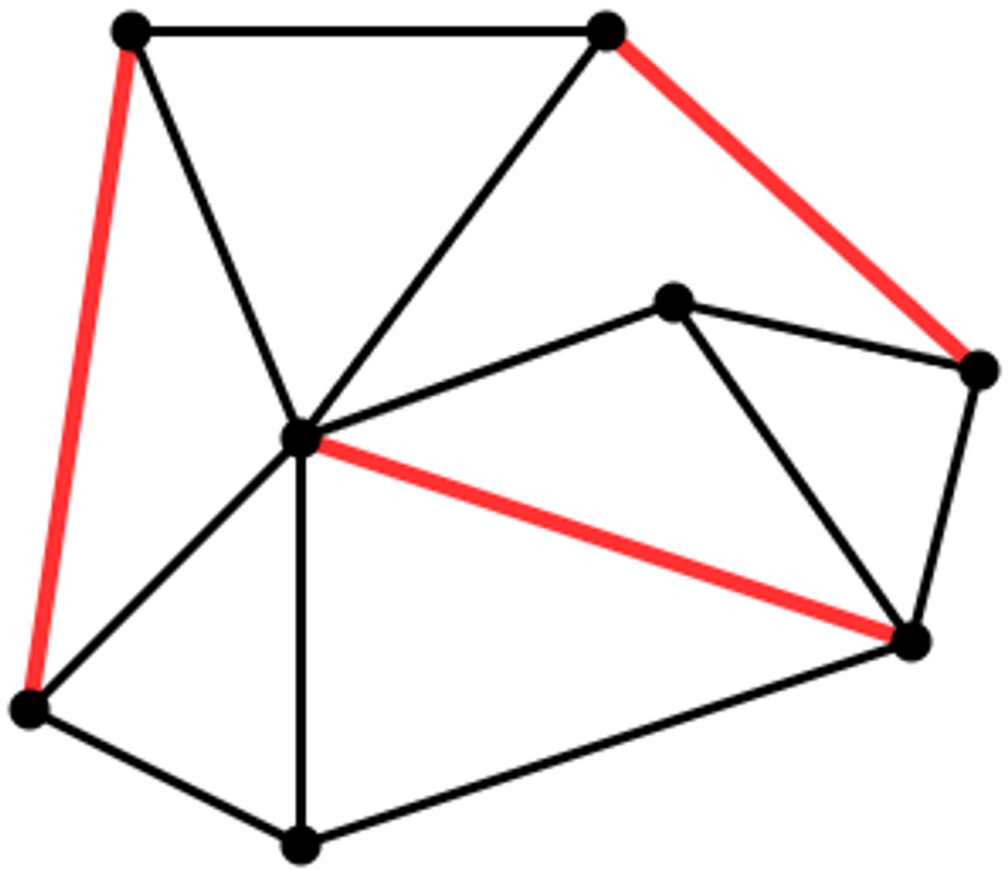
Step1 Find a maximal matching M



Classical Kernelization Suitable for Quantum Algo

Step1 Find a maximal matching M

if $|M| > k$: then No instance



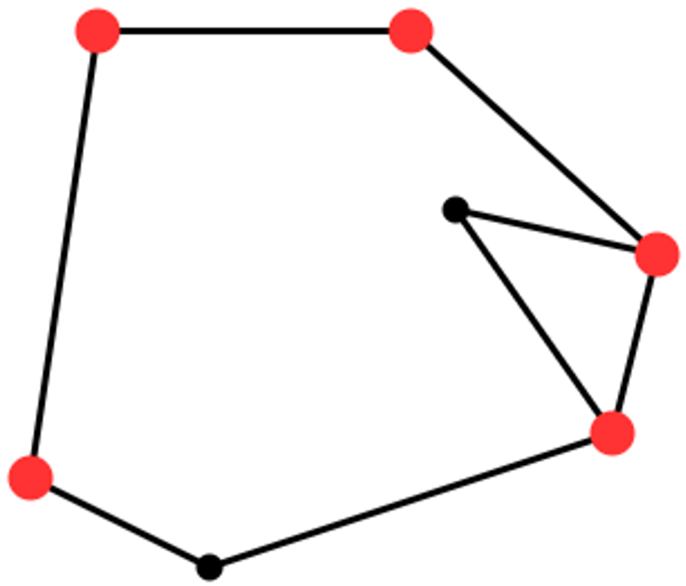
Classical Kernelization Suitable for Quantum Algo

Step1 Find a maximal matching M

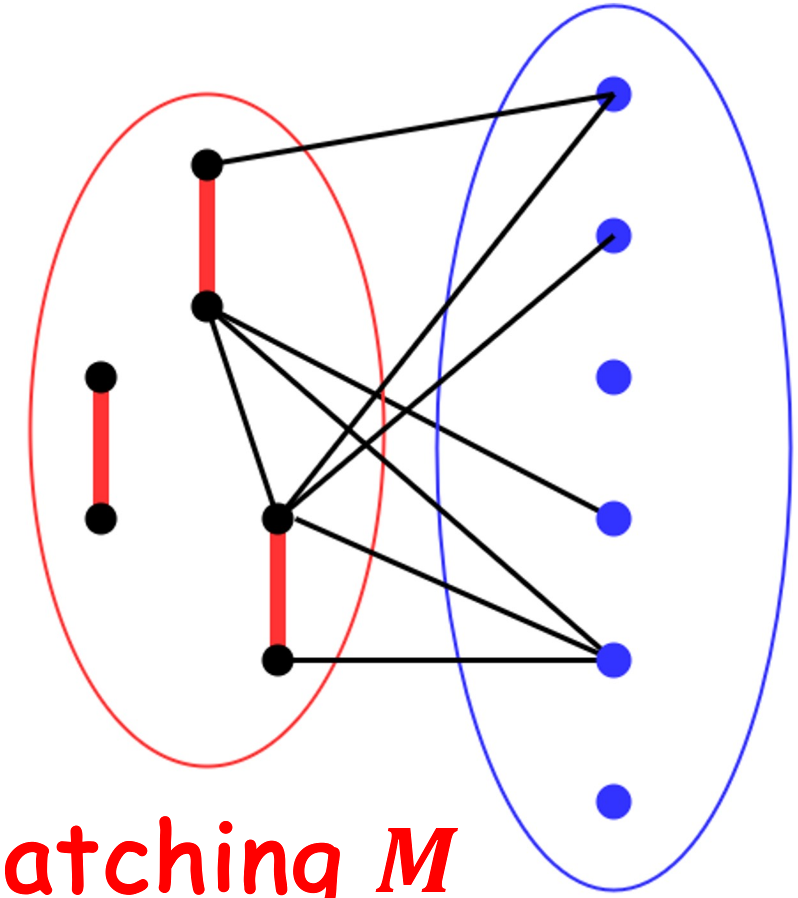
if $|M| > k$: then No instance

Step2 Apply Rule 2 only for endpoints of edges in M

Rule 2. If G has a vertex v of degree at least $k+1$, then $(G,k) \rightarrow (G-v, k-1)$



Crucial Observation



maximal matching M

independent set

☞ All edges touch an endpoint of an edge in M !

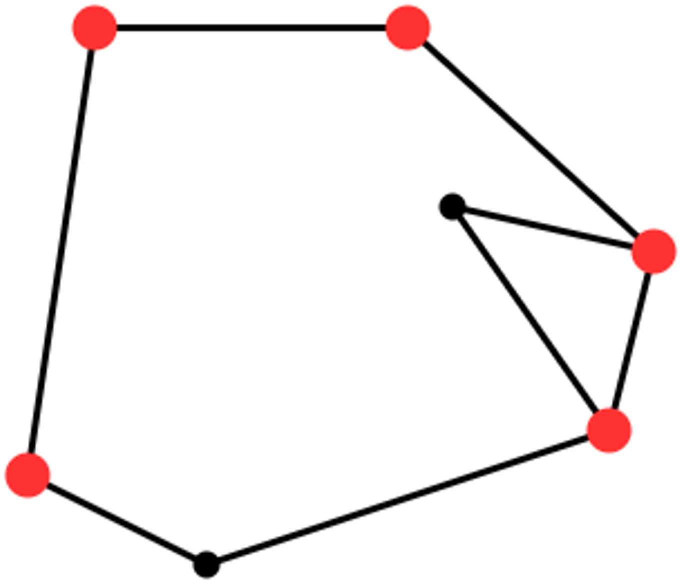
Classical Kernelization Suitable for Quantum Algo

Step1 Find a maximal matching M

if $|M| > k$: then No instance

Step2 Apply Rule 2 only for endpoints of edges in M

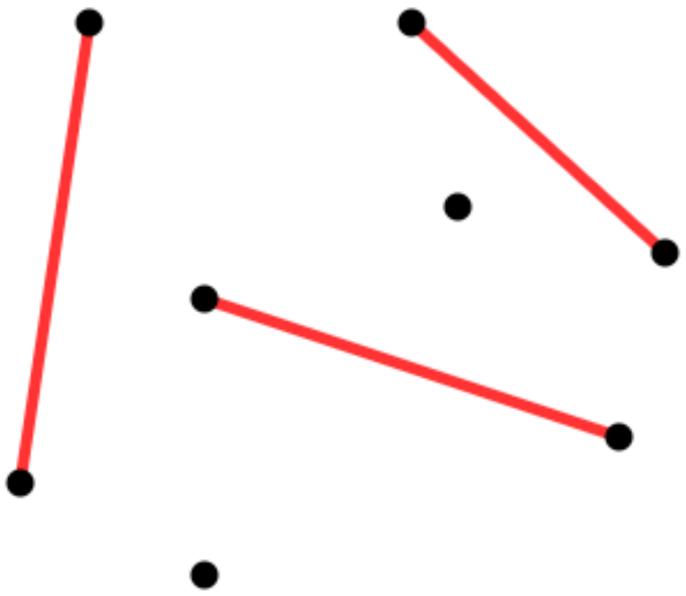
Rule 2. If G has a vertex v of degree at least $k+1$, then $(G,k) \rightarrow (G-v, k-1)$



Lem: After Step1 and 2, $|E(G)| \leq 2k^2$

Quantum Query Kernelization

Step1 Find a matching of size at least $k+1$ or a maximal matching of size at most k



Lem: Step1 uses $O(\sqrt{kn})$ queries

Quantum Query Kernelization

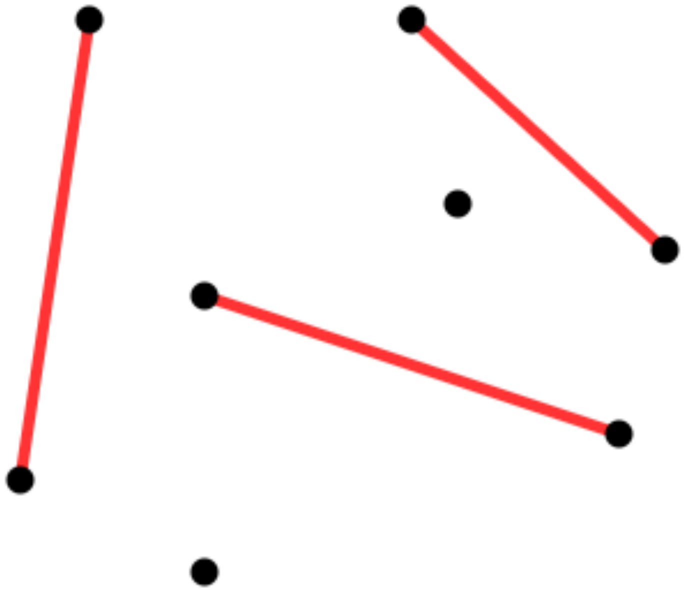
a matching of size at least $k+1$

Step1 Find

or

a maximal matching of siz

No instance



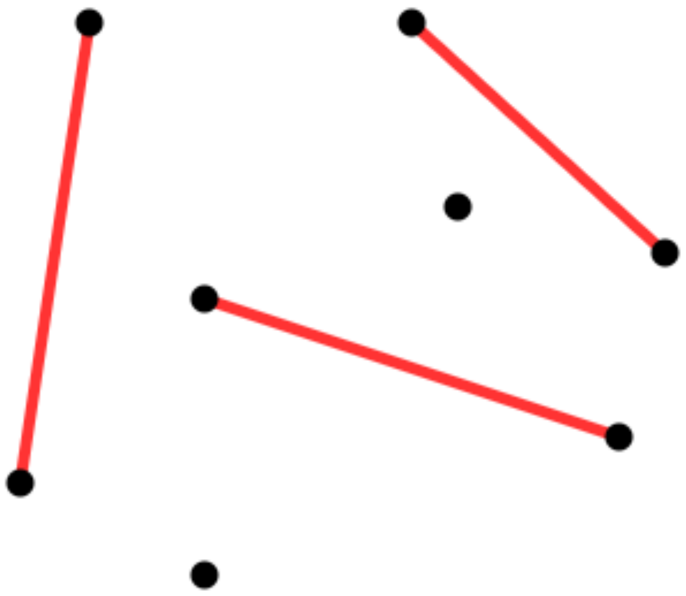
Lem: Step1 uses $O(\sqrt{kn})$ queries

Quantum Query Kernelization

a matching of size at least $k+1$

Step1 Find or

a maximal matching M of size at most k



Lem: Step1 uses $O(\sqrt{kn})$ queries

Quantum Query Kernelization

Step1 Find a matching of size at least $k+1$

or

a maximal matching M of size at most k

Step2 For each $v \in V(M)$:

all endpoints of edges in M

Quantum Query Kernelization

Step1 Find a matching of size at least $k+1$

or

a maximal matching $\boldsymbol{M}$ of size at most k

Step2 For each $\boldsymbol{v} \in \boldsymbol{V}(\boldsymbol{M})$: all endpoints of edges in $\boldsymbol{M}$

if (degree of v) $> \boldsymbol{k}$: then remove $\boldsymbol{v}$, $\boldsymbol{k} \leftarrow \boldsymbol{k} - \boldsymbol{1}$

else: find all edges incident to v

Quantum Query Kernelization

a matching of size at least $k + 1$

Step1 Find

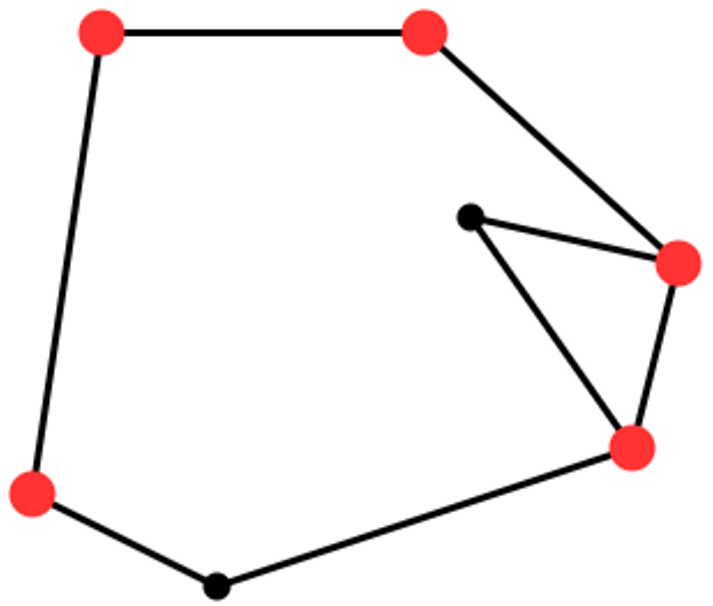
or

a maximal matching M of size at most k

Step2 For each $v \in V(M)$: all endpoints of edges in M

if (degree of v) $> k$: then remove v, $k \leftarrow k - 1$

else: find all edges incident to v



Quantum Query Kernelization

a matching of size at least $k+1$

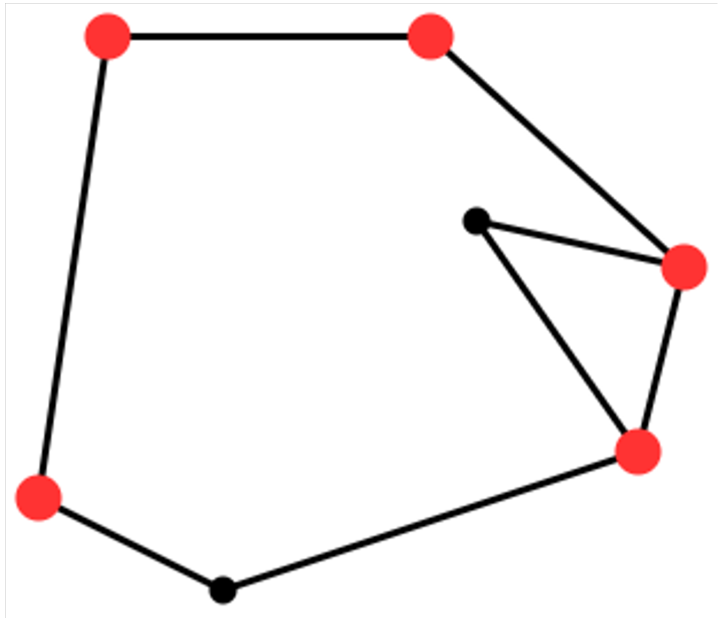
Step1 Find or

a maximal matching $\boldsymbol{M}$ of size at most k

Step2 For each $\boldsymbol{v} \in \boldsymbol{V}(\boldsymbol{M})$:

if (degree of v) $> k$: then remove v, $k \leftarrow k - 1$

else: find all edges incident to v

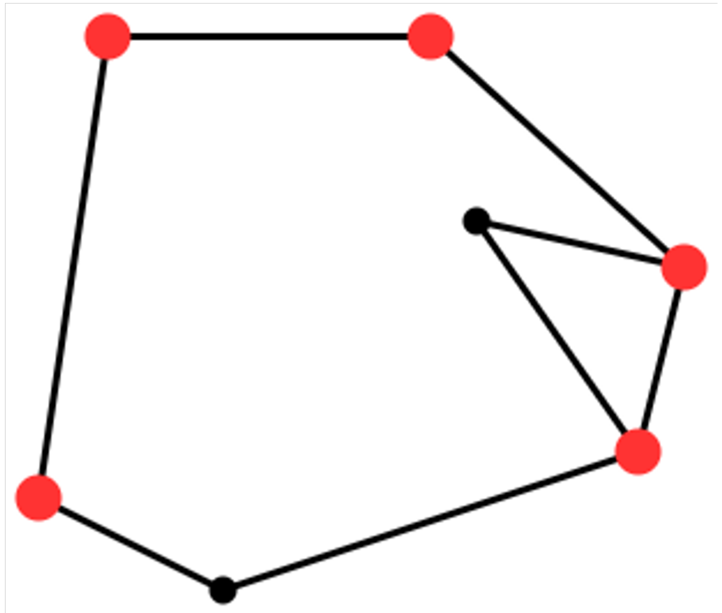


Obtain an equivalent instance as a bit string!

Quantum Query Kernelization

Step1 Find a matching of size at least $k+1$
or
a maximal matching M of size at most k

Step2 For each $v \in V(M)$:
if (degree of v) $> k$: then remove v, $k \leftarrow k-1$
else: find all edges incident to v



Lem:
Step2 uses $O(k^{3/2}\sqrt{n})$ queries

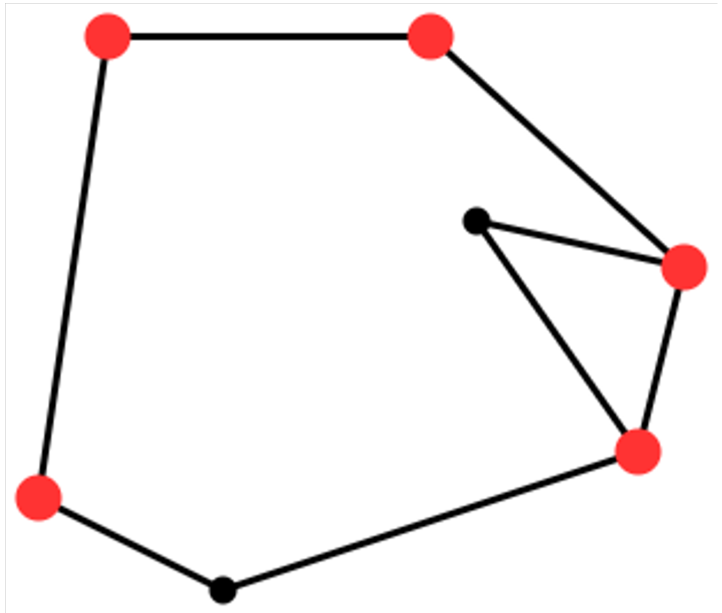
Quantum Query Kernelization

Step1 Find a matching of size at least $k+1$ or a maximal matching M of size at most k ⟸ $O(\sqrt{kn})$ queries

Step2 For each $v \in V(M)$: ⟸ $O(k^{3/2}\sqrt{n})$ queries

if (degree of v) $> k$: then remove v, $k \leftarrow k - 1$

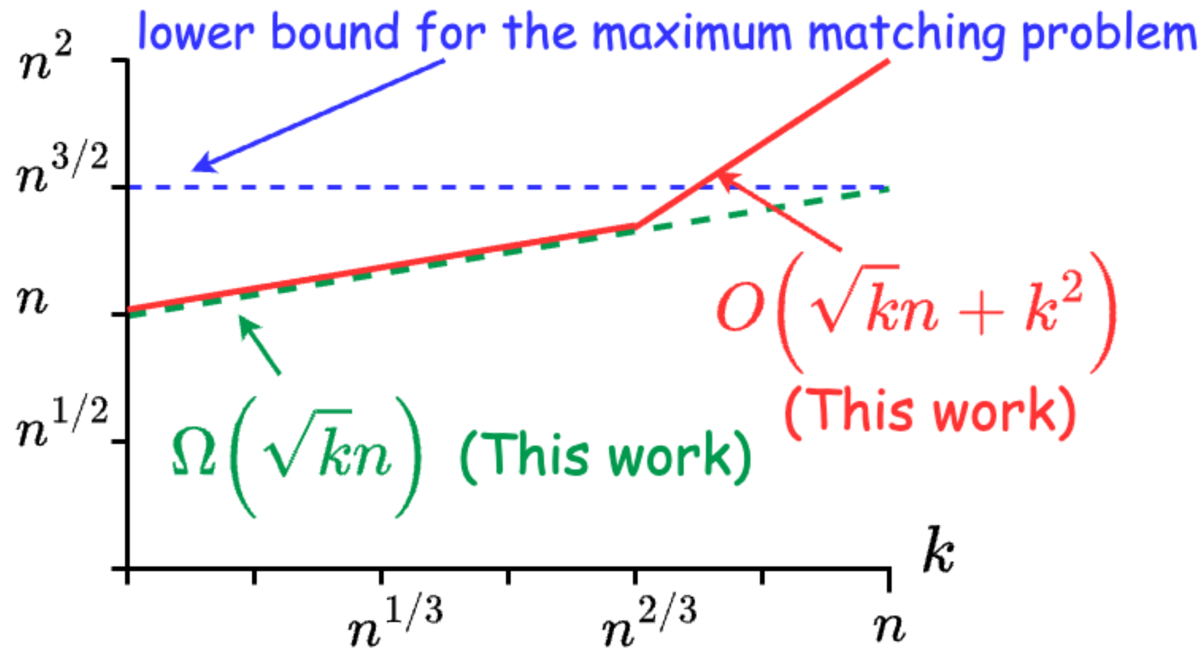
else: find all edges incident to v



Our Contribution 2. Parameterized Quantum Query Complexity for Matching

Thm.
Quantum Query Complexity to find a matching of size at least k

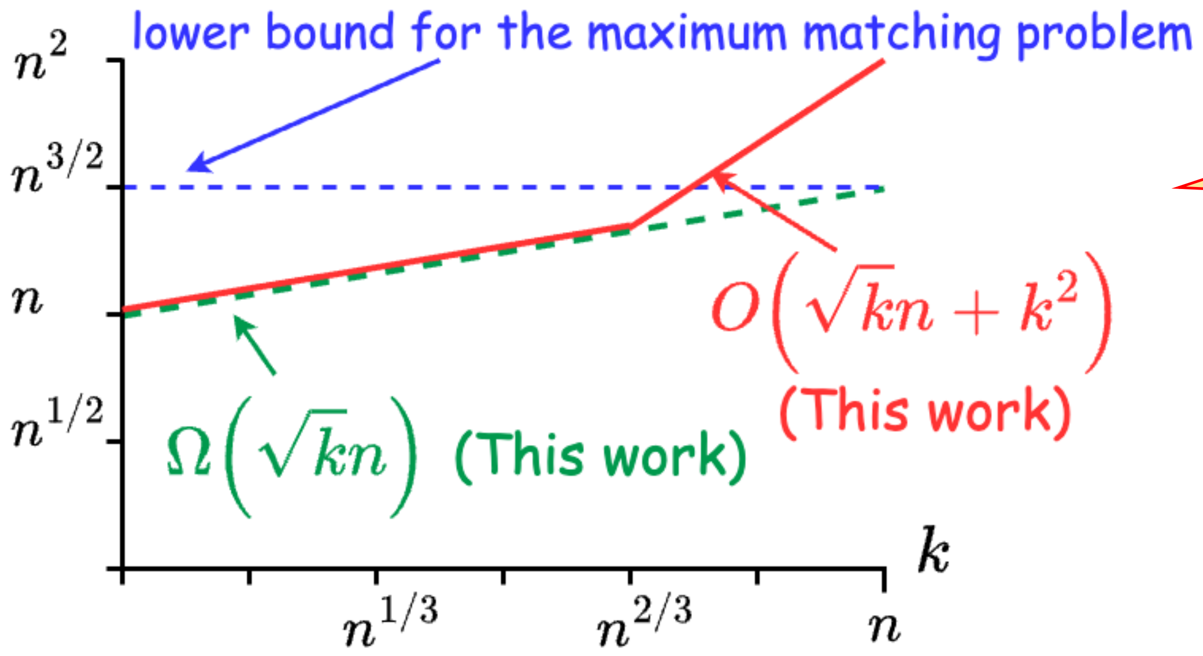
Upper Bound : $O(\sqrt{k}n + k^2)$
Lower Bound : $\Omega(\sqrt{k}n)$



Our Contribution 2. Parameterized Quantum Query Complexity for Matching

Thm.
Quantum Query Complexity to find a matching of size at least k

Upper Bound : $O(\sqrt{k}n + k^2)$
Lower Bound : $\Omega(\sqrt{k}n)$



Optimal complexity
$\Theta(\sqrt{k}n)$ when $k = O(n^{2/3})$

Our Contribution 2. Parameterized Quantum Query Complexity for Matching

Thm.

Quantum Query Complexity to find a matching of size at least k

Upper Bound : $O(\sqrt{k}n + k^2)$

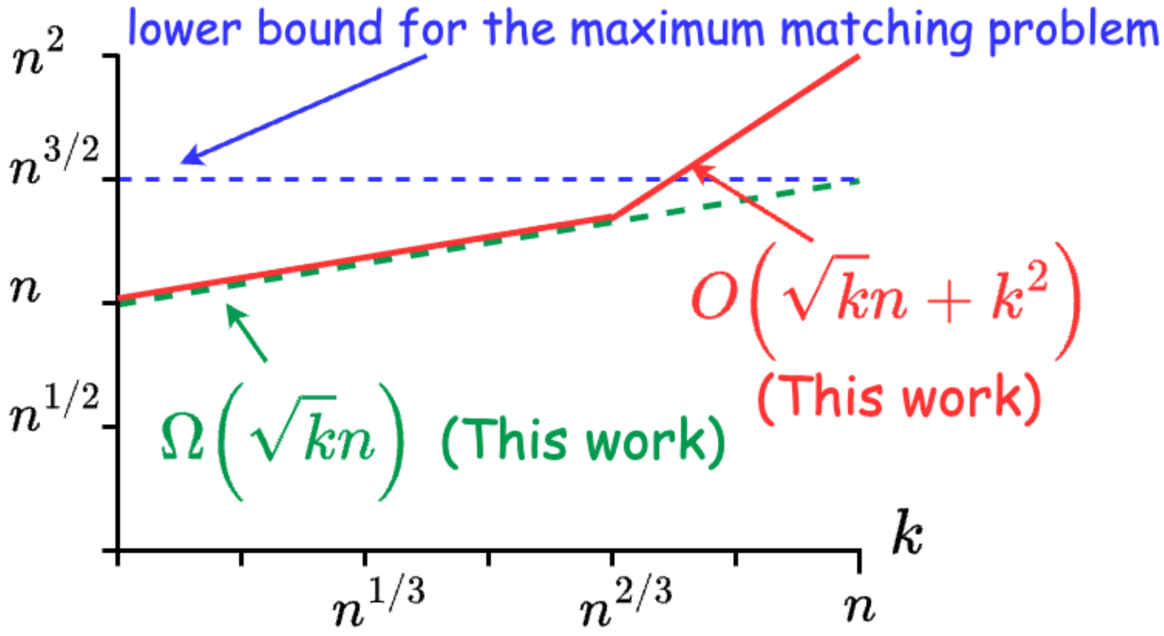
Lower Bound : $\Omega(\sqrt{k}n)$

Significance

- UB $O(n^{7/4})$ [Kimmel-Witter '21], LB $\Omega(n^{3/2})$ [Zhang '04] were only known for maximum matching
- Consider Parameterized ver.

Technique

- augmenting paths
- quantum query kernelization idea



Quantum Query Algo for k-matching

a matching of size at least $k+1$

Step1 Find or ⟸ $O(\sqrt{kn})$ queries

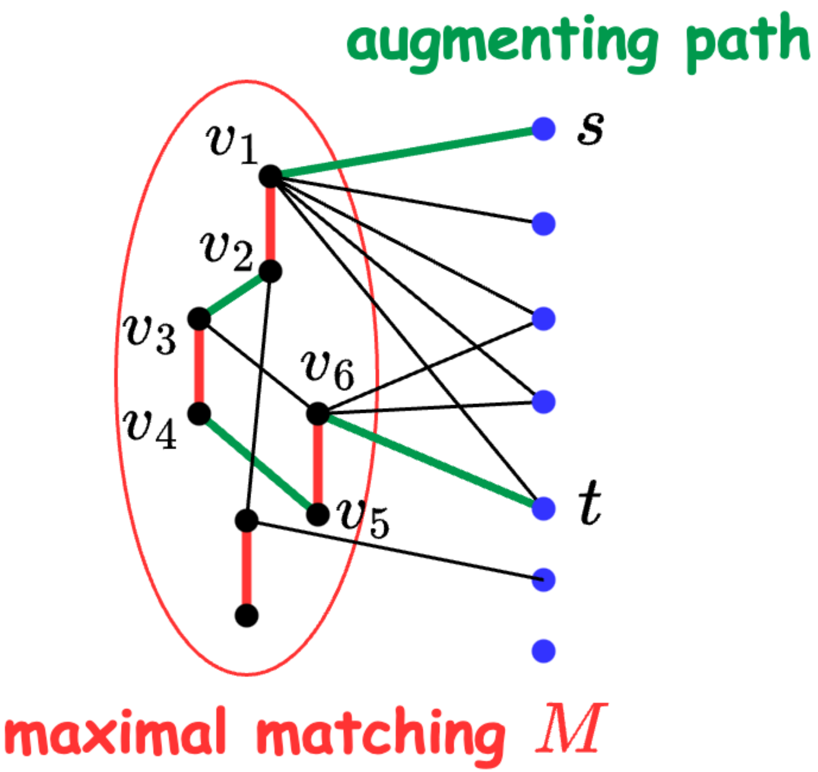
a maximal matching of size at most k

Quantum Query Algo for k-matching

Step1 Find a matching of size at least $k+1$ or a maximal matching M of size at most k ⟸ $O(\sqrt{kn})$ queries

Step2 Repeatedly find an augmenting path and augment along it

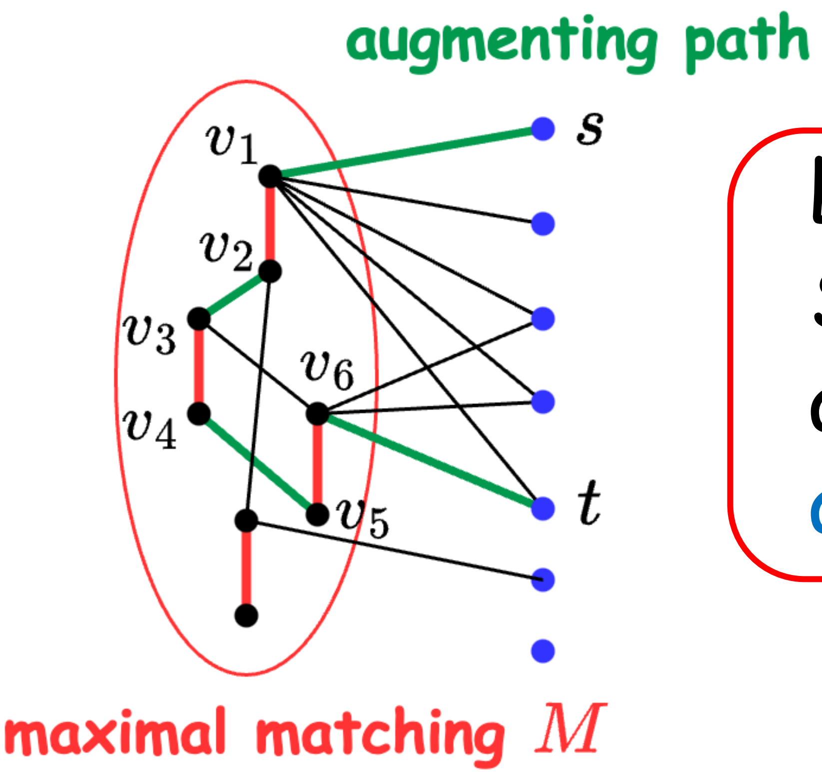
$|M|$ increases by 1 !



Quantum Query Algo for k-matching

Step1 Find a matching of size at least $k+1$ or a maximal matching M of size at most k ⟸ $O(\sqrt{kn})$ queries

Step2 Repeatedly find an augmenting path and augment along it

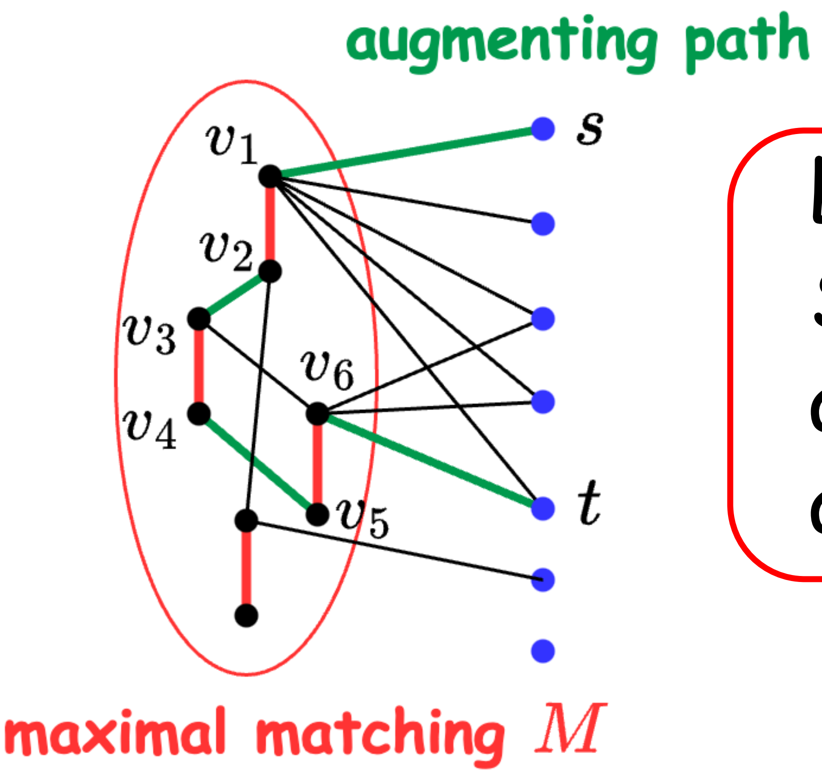


Lem:
Step2 uses $O(k^2)$ queries + amoritized $O(\sqrt{n})$ queries per one augmentation

Quantum Query Algo for k-matching

Step1 Find a matching of size at least $k+1$ or a maximal matching M of size at most k ⟸ $O(\sqrt{kn})$ queries

Step2 Repeatedly find an augmenting path and augment along it



Lem:
Step2 uses $O(k^2)$ queries + amoritized $O(\sqrt{n})$ queries per one augmentation

$O(k^2 + k\sqrt{n})$ queries

Conclusion

- Optimal Parameterized Quantum Query Complexities for vertex cover and matching when the parameters are small.

Message

☞ By making smart use of classical techniques such as kernelization, we can improve quantum query complexities !