Quantum algorithms for learning hidden graphs

Changpeng Shao

School of Mathematics, University of Bristol, UK
joint work with Ashley Montanaro
arXiv:2011.08611

23, February 2021

Background

Quantum computers can solve certain problems much faster than classical computers (Integer factorization [Shor, 1994], Searching [Grover, 1996]).

Background

Quantum computers can solve certain problems much faster than classical computers (Integer factorization [Shor, 1994], Searching [Grover, 1996]). They can learn unknown objects using fewer queries than their classical counterparts.

Background

Quantum computers can solve certain problems much faster than classical computers (Integer factorization [Shor, 1994], Searching [Grover, 1996]). They can learn unknown objects using fewer queries than their classical counterparts.

Example 1 (Bernstein-Vazirani algorithm, 1992)

$f : \{0,1\}^n \to \{0,1\}$ is promised to be $f(x) = x \cdot a$ for some unknown a. Given an oracle to implement f, find a.

Quantum vs Classical = 1 vs n.

- It proves an oracle separation between BQP and BPP.
- It is a subroutine of many useful quantum algorithms.
  [Bravyi, Gosset, Robert, 2018], [Lee, Santha, Zhang, 2021],...

Background

Example 2 (Combinatorial group testing (Belovs, 2013))

Assume $A \subseteq [n]$ *is of size* k. *For any* $S \subseteq [n]$, *define*

$$f_A(S) = \begin{cases} 1, & \text{if } A \cap S \neq \emptyset \\ 0, & \text{otherwise} \end{cases}$$

Given an oracle to implement f_A, *find* A.

Quantum vs Classical $= \sqrt{k}$ *vs* $k \log(n/k)$.

- Dates back to 1943. It was proposed as a means of identifying and rejecting syphilitic soldiers in the US military.
- A main technique of our paper.

Problem statement

Problem 1 (Learning a hidden graph)

Given a unknown graph $G = (V, E)$ with an oracle to query V, using fewer queries to determine this graph, i.e., determine E.

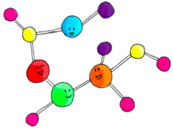
Problem statement

Problem 1 (Learning a hidden graph)

Given a unknown graph $G = (V, E)$ with an oracle to query V, using fewer queries to determine this graph, i.e., determine E.

Motivated by wide applications in molecular biology:

- vertices: atoms
- edges: reactions



- queries: experiments of putting a set of atoms together in a test tube and determining whether a reaction occurs

Different query models

Local queries:

1. **Edge-existence query**
   For any $u, v \in V$, determine if $(u, v) \in E$.
2. **Degree query**
   For any $u \in V$, return the degree of u.
3. **Neighbor query**
   For any $u \in V, j \in [n]$, return the j-th neighbor of u if exists.

Different query models

Local queries:

1. **Edge-existence query**
   For any $u, v \in V$, determine if $(u, v) \in E$.
2. **Degree query**
   For any $u \in V$, return the degree of u.
3. **Neighbor query**
   For any $u \in V, j \in [n]$, return the j-th neighbor of u if exists.

Global queries:

1. **OR query** (aks independent set query, edge-detection query)
   For any $S \subseteq V$, determines if S contains any edges.
2. **Subset query**
   For any $S \subseteq V \times V$, determines if S contains any edges.
3. **Additive query** (aks quantitative query, edge counting query)
   For any $S \subseteq V$, returns the number of edges in S.

Queries considered in our paper

For certain problems, global queries are exponentially efficient than local queries, e.g., [Beame, Har-Peled, Ramamoorthy, Rashtchian, Sinha, 2017], [Chen, Levy, Waingarten, 2020],...

Queries considered in our paper

For certain problems, global queries are exponentially efficient than local queries, e.g., [Beame, Har-Peled, Ramamoorthy, Rashtchian, Sinha, 2017], [Chen, Levy, Waingarten, 2020],...

We will focus on

1. **OR query**
   For any $S \subseteq V$, determines if S contains any edges.
2. **Parity query** (weaker than additive query)
   For any $S \subseteq V$, returns the parity of the number of edges in S.
3. **Graph state** (no classical counterpart, related to parity query)
   Given access to $|G\rangle = \prod_{(i,j)\in E} CZ_{ij}|+\rangle^{\otimes n}$.

OR query model (classical results)

For special graphs ($n = \#$ vertices):

- Matching: $O(n \log n)$ [Alon, Beigel, Kasif, Rudich, Sudakov, 2004]
- Hamiltonian cycle: $O(n \log n)$ [Grebinski, Kucherov, 1997]
- Star and clique: $O(n)$ [Bouvel, Grebinski, Kucherov, 2005]

For graphs with m-edges:

- m is known: $O(m \log n)$ [Angluin, Chen, 2008]
- m is unknown: $O(m \log n + \sqrt{m}(\log n)(\log \overset{k}{\cdots} \log n))$, where k can be any constant. [Hasan, Bshouty, 2019]

OR query model (quantum results)

In the table, $m = \#$ edges, $n = \#$ vertices:

	Quantum	Classical	
All graphs	$\Theta(n^2)$	$\Theta(n^2)$	No speedup

OR query model (quantum results)

In the table, $m = \#$ edges, $n = \#$ vertices:

	Quantum	Classical	
All graphs	$\Theta(n^2)$	$\Theta(n^2)$	No speedup
m edges	$O(m \log(m \log n))$ $\Omega(m)$	$\Omega(m \log \frac{n^2}{m})$	Speedup when $m \ll n$

OR query model (quantum results)

In the table, $m = \#$ edges, $n = \#$ vertices:

	Quantum	Classical	
All graphs	$\Theta(n^2)$	$\Theta(n^2)$	No speedup
m edges	$O(m \log(m \log n))$ $\Omega(m)$	$\Omega(m \log \frac{n^2}{m})$	Speedup when $m \ll n$
Matching Cycle Star k-vertex clique	$O(m^{3/4}),\ \Omega(m^{1/2})$ $O(m^{3/4}),\ \Omega(m^{1/2})$ $\Theta(\sqrt{m})$ $\Theta(\sqrt{k})$	$\Omega(m \log \frac{n}{m})$ $\Omega(m \log \frac{n}{m})$ $\Omega(m \log \frac{n}{m})$ $\Omega(k \log \frac{n}{k})$	Polynomial speedups

OR query model (quantum results)

In the table, $m = \#$ edges, $n = \#$ vertices:

	Quantum	Classical	
All graphs	$\Theta(n^2)$	$\Theta(n^2)$	No speedup
m edges	$O(m \log(m \log n))$ $\Omega(m)$	$\Omega(m \log \frac{n^2}{m})$	Speedup when $m \ll n$
Matching	$O(m^{3/4})$, $\Omega(m^{1/2})$	$\Omega(m \log \frac{n}{m})$	Polynomial speedups
Cycle	$O(m^{3/4})$, $\Omega(m^{1/2})$	$\Omega(m \log \frac{n}{m})$	
Star	$\Theta(\sqrt{m})$	$\Omega(m \log \frac{n}{m})$	
k-vertex clique	$\Theta(\sqrt{k})$	$\Omega(k \log \frac{n}{k})$	

- At most polynomial speedups.
- The classical lower bounds are obtained by information theoretical arguments.

Additive query model (classical results)

For special graphs:

- Matching: $O(n)$ [Grebinski, Kucherov, 2000]
- Hamiltonian cycle: $O(n)$ [Bouvel, Grebinski, Kucherov, 2005]
- Star and clique: $O(n/\log n)$ [Bouvel, Grebinski, Kucherov, 2005]

For graphs with m-edges:

- $O(m(\log n)/\log m)$ [Bshouty, Mazzawi, 2011]

Parity query model (quantum results)

In the table, $m = \#$ edges, $n = \#$ vertices:

	Quantum	Classical	
All graphs	$\Theta(n)$	$\Theta(n^2)$	Quadratic speedup
m edges	$O(\sqrt{m \log m})$	$\Omega(m \log \frac{n^2}{m})$	

Parity query model (quantum results)

In the table, $m = \#$ edges, $n = \#$ vertices:

	Quantum	Classical	
All graphs	$\Theta(n)$	$\Theta(n^2)$	Quadratic speedup
m edges	$O(\sqrt{m \log m})$	$\Omega(m \log \frac{n^2}{m})$	
Degree d	$O(d \log \frac{m}{d})$	$\Omega(nd \log \frac{n}{d})$	
Matching	$O(\log m)$	$\Omega(m \log \frac{n}{m})$	
Cycle	$O(\log m)$	$\Omega(m \log \frac{n}{m})$	Exponential speedups
Star	$O(1)$	$\Omega(m \log \frac{n}{m})$	
k-vertex clique	$O(1)$	$\Omega(k \log \frac{n}{k})$	

Parity query model (quantum results)

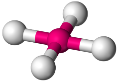
In the table, $m = \#$ edges, $n = \#$ vertices:

	Quantum	Classical	
All graphs	$\Theta(n)$	$\Theta(n^2)$	Quadratic speedup
m edges	$O(\sqrt{m \log m})$	$\Omega(m \log \frac{n^2}{m})$	
Degree d	$O(d \log \frac{m}{d})$	$\Omega(nd \log \frac{n}{d})$	
Matching	$O(\log m)$	$\Omega(m \log \frac{n}{m})$	
Cycle	$O(\log m)$	$\Omega(m \log \frac{n}{m})$	Exponential speedups
Star	$O(1)$	$\Omega(m \log \frac{n}{m})$	
k-vertex clique	$O(1)$	$\Omega(k \log \frac{n}{k})$	

- For graph state model, the only difference is learning a graph of m edges, the cost is $O(m \log \frac{n^2}{m})$.

Example 1: Learning stars by OR query

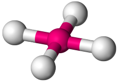
- Suppose the center is i, the edges are $(i, j), j \in A$.



Example 1: Learning stars by OR query

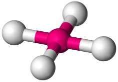
- Suppose the center is i, the edges are $(i, j), j \in A$.
- It is equivalent to learn $f = x_i \wedge (\vee_{j \in A} x_j)$.

  Let $S \subseteq [n]$, then OR$(S) = 1$ iff $i, j \in S$ for some j iff $f(S) = 1$.



Example 1: Learning stars by OR query

- Suppose the center is i, the edges are $(i, j), j \in A$.



- It is equivalent to learn $f = x_i \wedge (\vee_{j \in A} x_j)$.

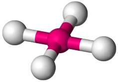
  Let $S \subseteq [n]$, then $\mathrm{OR}(S) = 1$ iff $i, j \in S$ for some j iff $f(S) = 1$.

- Consider the following procedure (Fourier sampling):

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \quad \mapsto \quad \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle$$

$$\mapsto \quad \frac{1}{2^n} \sum_{x,y \in \{0,1\}^n} (-1)^{f(x) + x \cdot y} |y\rangle$$

Example 1: Learning stars by OR query

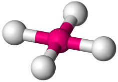


- Suppose the center is i, the edges are $(i,j), j \in A$.
- It is equivalent to learn $f = x_i \wedge (\vee_{j \in A} x_j)$.
  Let $S \subseteq [n]$, then $\mathrm{OR}(S) = 1$ iff $i, j \in S$ for some j iff $f(S) = 1$.
- Consider the following procedure (Fourier sampling):

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle$$
$$\mapsto \frac{1}{2^n} \sum_{x,y \in \{0,1\}^n} (-1)^{f(x)+x \cdot y} |y\rangle$$

- The coefficient of $y_i = 1, y_j = 0 \ (j \neq i)$ equals $1 - 2^{1-m}$. Perform measurements, with probability $(1 - 2^{1-m})^2$ we obtain the center i.

Example 1: Learning stars by OR query



- Suppose the center is i, the edges are $(i, j), j \in A$.
- It is equivalent to learn $f = x_i \wedge (\vee_{j \in A} x_j)$.
  Let $S \subseteq [n]$, then $\mathrm{OR}(S) = 1$ iff $i, j \in S$ for some j iff $f(S) = 1$.
- Consider the following procedure (Fourier sampling):

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle$$
$$\mapsto \frac{1}{2^n} \sum_{x,y \in \{0,1\}^n} (-1)^{f(x)+x \cdot y} |y\rangle$$

- The coefficient of $y_i = 1, y_j = 0$ $(j \neq i)$ equals $1 - 2^{1-m}$. Perform measurements, with probability $(1 - 2^{1-m})^2$ we obtain the center i.
- It remains to learn $f' = \vee_{j \in A} x_j$. This is a group testing problem (Belovs' algorithm). (Overall cost: $\Theta(\sqrt{|A|})$)

Example 2: Learning stars by Parity query

- Suppose the center is i, the edges are $(i, j), j \in A$.

Example 2: Learning stars by Parity query

- Suppose the center is i, the edges are $(i, j), j \in A$.
- $f(x) = \sum_{i,j} x_i x_j \mod 2 = x_i \sum_{j \in A} x_j \mod 2$.

Example 2: Learning stars by Parity query

- Suppose the center is i, the edges are $(i, j), j \in A$.
- $f(x) = \sum_{i,j} x_i x_j \mod 2 = x_i \sum_{j \in A} x_j \mod 2$.
- By Fourier sampling, we obtain

$$
\begin{aligned}
&\frac{1}{\sqrt{2}}|0, \ldots, 0\rangle|+\rangle|0, \ldots, 0\rangle \\
+&\frac{1}{\sqrt{2}}|[1 \in A], \ldots, [i-1 \in A]\rangle|-\rangle|[i+1 \in A], \ldots, [n \in A]\rangle
\end{aligned}
$$

The $|\pm\rangle$ is in the i-th qubit, $[j \in A] = 1$ if $j \in A$ and 0 otherwise. (Overall cost: $O(1)$)

Example 3: Learning graphs of m edges by OR query

The idea comes from [Angluin, Chen, 2008]

1. Decompose $V = V_1 \cup \cdots \cup V_k$ (disjoint union, i.e., k-coloring), such that each V_i includes no edges (hope: k small).

2. Find all the edges between V_i, V_j.

Example 3: Learning graphs of m edges by OR query

The idea comes from [Angluin, Chen, 2008]

1. Decompose $V = V_1 \cup \cdots \cup V_k$ (disjoint union, i.e., k-coloring), such that each V_i includes no edges (hope: k small).
2. Find all the edges between V_i, V_j.

A p-random set S of V is obtained by including each vertex independently with probability p. Then

$$\text{Prob}[S \text{ includes no edges}] \geq 1 - mp^2.$$

Example 3: Learning graphs of m edges by OR query

The idea comes from [Angluin, Chen, 2008]

1. Decompose $V = V_1 \cup \cdots \cup V_k$ (disjoint union, i.e., k-coloring), such that each V_i includes no edges (hope: k small).

2. Find all the edges between V_i, V_j.

A p-random set S of V is obtained by including each vertex independently with probability p. Then

$$\text{Prob}[S \text{ includes no edges}] \geq 1 - mp^2.$$

Choose $p = 0.1/\sqrt{m}$.

1. With probability ≥ 0.99, we can find V_1.
2. In $V - V_1$, we can similarly find V_2, and so on.
3. $k \approx \sqrt{m} \log n$ (optimal, e.g. complete graph).

Find the edges between V_i, V_j

Lemma 1

Assume there are m_{ij} edges between V_i and V_j. Then the edges can be identified with $O(m_{ij})$ OR queries.

Find the edges between V_i, V_j

Lemma 1

Assume there are m_{ij} edges between V_i and V_j. Then the edges can be identified with $O(m_{ij})$ OR queries.

Proof.

Suppose $\{x_1, \ldots, x_p\} \subseteq V_i$ are connected to $\{y_1, \ldots, y_q\} \subseteq V_j$.

- Equivalent to learn $f = x_1 f_1 \vee \cdots \vee x_p f_p$, where $f_1, \ldots, f_p$ are OR functions of $y_1, \ldots, y_q$.
- Set $V_j = 1$, then $f = x_1 \vee \cdots \vee x_p$. (group testing)
- Set $x_i = 1, x_j = 0$ $(j \neq i)$, then learn f_i. (group testing)

□

Find the edges between V_i, V_j

Lemma 1

Assume there are m_{ij} edges between V_i and V_j. Then the edges can be identified with $O(m_{ij})$ OR queries.

Proof.

Suppose $\{x_1, \ldots, x_p\} \subseteq V_i$ are connected to $\{y_1, \ldots, y_q\} \subseteq V_j$.

- Equivalent to learn $f = x_1 f_1 \vee \cdots \vee x_p f_p$, where $f_1, \ldots, f_p$ are OR functions of $y_1, \ldots, y_q$.
- Set $V_j = 1$, then $f = x_1 \vee \cdots \vee x_p$. (group testing)
- Set $x_i = 1, x_j = 0$ $(j \neq i)$, then learn f_i. (group testing)

□

If the graph has max degree $O(1)$, the result can be improved to $O(\sqrt{m_{ij}} \log m_{ij})$.

Learn all the edges

Since $k = \sqrt{m} \log n$, there are $O(k^2)$ pairs. So it totally costs $O(m \log^2 n)$. This is worse than the classical result $O(m \log n)$. There is a way to reduce the dependence on k to linear.

Learn all the edges

Since $k = \sqrt{m} \log n$, there are $O(k^2)$ pairs. So it totally costs $O(m \log^2 n)$. This is worse than the classical result $O(m \log n)$. There is a way to reduce the dependence on k to linear.

Lemma 2

Assume that A and B are two disjoint sets of V with m_A, m_B known edges respectively. Suppose there are m_{AB} edges between A and B. Then the edges can be identified using $O(m_{AB} + m_A + m_B)$ OR queries.

Learn all the edges

Since $k = \sqrt{m} \log n$, there are $O(k^2)$ pairs. So it totally costs $O(m \log^2 n)$. This is worse than the classical result $O(m \log n)$. There is a way to reduce the dependence on k to linear.

Lemma 2

Assume that A and B are two disjoint sets of V with m_A, m_B known edges respectively. Suppose there are m_{AB} edges between A and B. Then the edges can be identified using $O(m_{AB} + m_A + m_B)$ OR queries.

Proof.

Fact: a graph of t edges can be $\lfloor \sqrt{2t} + 1 \rfloor$ colored.
Learn the edges of each pair of color classes by Lemma 1. □

Learn all the edges

Theorem 1

Suppose the graph G has m edges, then there is a quantum algorithm that learns all the edges using

$$O(m\log(\sqrt{m}\log n) + \sqrt{m}\log n)$$

OR queries.

Learn all the edges

Theorem 1

Suppose the graph G has m edges, then there is a quantum algorithm that learns all the edges using

$$O(m \log(\sqrt{m} \log n) + \sqrt{m} \log n)$$

OR queries.

Proof.

1. Learn the edges between V_{2i-1}, V_{2i}. Then combine them.
2. Apply the same idea to the new $k/2$ subsets.
3. $k = \sqrt{m} \log n$.

□

Learn all the edges

Theorem 1

Suppose the graph G has m edges, then there is a quantum algorithm that learns all the edges using

$$O(m \log(\sqrt{m} \log n) + \sqrt{m} \log n)$$

OR queries.

Proof.

1. Learn the edges between V_{2i-1}, V_{2i}. Then combine them.
2. Apply the same idea to the new $k/2$ subsets.
3. $k = \sqrt{m} \log n$.

□

- If the graph has max degree $O(1)$ and $O(1)$-colorable, the result is improved to $O(m^{3/4}(\log m)\sqrt{\log n} + \sqrt{m} \log n)$.

Lower bounds

Theorem 2

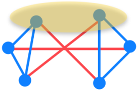
Let G be an arbitrary graph of n vertices. Then any quantum algorithm that learns G must make $\Omega(n^2)$ OR queries.

Lower bounds

Theorem 2

Let G be an arbitrary graph of n vertices. Then any quantum algorithm that learns G must make $\Omega(n^2)$ OR queries.

Proof.



Blue edges are known, there are $k \leq n^2$ unknown red edges.

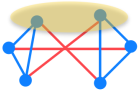
quantum search: finding k edges costs $\Theta(\sqrt{n^2 k})$ queries. □

Lower bounds

Theorem 2

Let G be an arbitrary graph of n vertices. Then any quantum algorithm that learns G must make $\Omega(n^2)$ OR queries.

Proof.



Blue edges are known, there are $k \leq n^2$ unknown red edges.

quantum search: finding k edges costs $\Theta(\sqrt{n^2 k})$ queries. □

As a corollary,

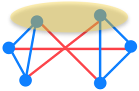
- Any quantum algorithm that learns an arbitrary graph with m edges must make $\Omega(m)$ queries.

Lower bounds

Theorem 2

Let G be an arbitrary graph of n vertices. Then any quantum algorithm that learns G must make $\Omega(n^2)$ OR queries.

Proof.



Blue edges are known, there are $k \leq n^2$ unknown red edges.

quantum search: finding k edges costs $\Theta(\sqrt{n^2 k})$ queries. □

As a corollary,

- Any quantum algorithm that learns an arbitrary graph with m edges must make $\Omega(m)$ queries.
- Any quantum algorithm that determines m exactly must make $\Omega(m)$ queries when $m = \Omega(n^2)$. quantum counting: compute $\tilde{m}$ such that $|m - \tilde{m}| \leq \epsilon m$ costs $\Theta(\frac{1}{\epsilon}\sqrt{\frac{n^2}{m}})$ queries. Choose $\epsilon \approx 1/m$.

Graph states and parity query

Let $G = (V, E)$ be a graph, then its graph state is defined as

$$|G\rangle = \prod_{(i,j)\in E} CZ_{ij}|+\rangle^{\otimes n}$$
$$= \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{\sum_{(i,j)\in E} x_i x_j} |x\rangle,$$

where $\sum_{(i,j)\in E} x_i x_j \mod 2$ is the parity query.

Graph states and parity query

Let $G = (V, E)$ be a graph, then its graph state is defined as

$$\begin{aligned} |G\rangle &= \prod_{(i,j)\in E} CZ_{ij}|+\rangle^{\otimes n} \\ &= \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{\sum_{(i,j)\in E} x_i x_j} |x\rangle, \end{aligned}$$

where $\sum_{(i,j)\in E} x_i x_j \mod 2$ is the parity query.

It is the unique state stabilized by the set of Pauli operators

$$\{X_v \prod_{w\in N(v)} Z_w : v \in V\},$$

where $N(v)$ denotes the set of vertices neighbouring v.

[Hein, Dür, Eisert, Raussendorf, Van den Nest, Briegel, 2006],
[Zhao, Pérez-Delgado, Fitzsimons, 2016],...

Bell sampling

Lemma 3 (Montanaro, 2017)

Let $|\psi\rangle$ *be a state of* n *qubits. Bell sampling applied to* $|\psi\rangle^{\otimes 2}$ *returns outcome* s *with probability*

$$\frac{|\langle\psi|\sigma_s|\psi^*\rangle|^2}{2^n},$$

where $|\psi^*\rangle$ *is the complex conjugate of* $|\psi\rangle$ *with respect to the computational basis, and* $\sigma_s = s_1 \otimes s_2 \otimes \cdots \otimes s_n$.

Bell sampling

Lemma 3 (Montanaro, 2017)

Let $|\psi\rangle$ *be a state of* n *qubits. Bell sampling applied to* $|\psi\rangle^{\otimes 2}$ *returns outcome* s *with probability*

$$\frac{|\langle\psi|\sigma_s|\psi^*\rangle|^2}{2^n},$$

where $|\psi^*\rangle$ *is the complex conjugate of* $|\psi\rangle$ *with respect to the computational basis, and* $\sigma_s = s_1 \otimes s_2 \otimes \cdots \otimes s_n$.

If $|G\rangle$ is a graph state, Bell sampling returns a uniformly random stabilizer of $|G\rangle$:

$$\prod_{v\in S} X_v \prod_{u\in N(v)} Z_u = \prod_{u\in[n]} X_u^{[u\in S]} Z_u^{|N(u)\cap S|}.$$

Bell sampling

Lemma 3 (Montanaro, 2017)

Let $|\psi\rangle$ be a state of n qubits. Bell sampling applied to $|\psi\rangle^{\otimes 2}$ returns outcome s with probability

$$\frac{|\langle\psi|\sigma_s|\psi^*\rangle|^2}{2^n},$$

where $|\psi^*\rangle$ is the complex conjugate of $|\psi\rangle$ with respect to the computational basis, and $\sigma_s = s_1 \otimes s_2 \otimes \cdots \otimes s_n$.

If $|G\rangle$ is a graph state, Bell sampling returns a uniformly random stabilizer of $|G\rangle$:

$$\prod_{v\in S} X_v \prod_{u\in N(v)} Z_u = \prod_{u\in[n]} X_u^{[u\in S]} Z_u^{|N(u)\cap S|}.$$

View S as a bit sting s, then it corresponds to $As \mod 2$.

Learning graphs from a family

Theorem 3

Let $\mathcal{F}$ be a family of graphs. Then, for any $G \in \mathcal{F}$, it can be identified by applying Bell sampling to $O(\log |\mathcal{F}|)$ copies of $|G\rangle$.

Learning graphs from a family

Theorem 3

Let $\mathcal{F}$ be a family of graphs. Then, for any $G \in \mathcal{F}$, it can be identified by applying Bell sampling to $O(\log |\mathcal{F}|)$ copies of $|G\rangle$.

Proof.

Generate k Bell samples, then we obtain boolean matrices B and AB. By the union bound, $\Pr_B[\exists C = A + A', CB = 0] \leq |\mathcal{F}|^2/2^k$. So to uniquely determine A, we choose $k = O(\log |\mathcal{F}|)$. □

Learning graphs from a family

Theorem 3

Let $\mathcal{F}$ be a family of graphs. Then, for any $G \in \mathcal{F}$, it can be identified by applying Bell sampling to $O(\log |\mathcal{F}|)$ copies of $|G\rangle$.

Proof.

Generate k Bell samples, then we obtain boolean matrices B and AB. By the union bound, $\Pr_B[\exists C = A + A', CB = 0] \leq |\mathcal{F}|^2/2^k$. So to uniquely determine A, we choose $k = O(\log |\mathcal{F}|)$. □

e.g. If G is a graph with at most m edges, it can be identified with $O(m \log(n^2/m))$ copies of $|G\rangle$.

Learning graphs from a family

Theorem 3

Let $\mathcal{F}$ be a family of graphs. Then, for any $G \in \mathcal{F}$, it can be identified by applying Bell sampling to $O(\log|\mathcal{F}|)$ copies of $|G\rangle$.

Proof.

Generate k Bell samples, then we obtain boolean matrices B and AB. By the union bound, $\Pr_B[\exists C = A + A', CB = 0] \leq |\mathcal{F}|^2/2^k$. So to uniquely determine A, we choose $k = O(\log|\mathcal{F}|)$. □

e.g. If G is a graph with at most m edges, it can be identified with $O(m\log(n^2/m))$ copies of $|G\rangle$.

By information-theoretic arguments, $\Omega(\log|\mathcal{F}|)$ is the lower bound to learn graphs in the classical setting. In the quantum setting, the lower bound is $\Omega(\sqrt{\log|\mathcal{F}|})$.

Learning bounded-degree graphs

Theorem 4 (Bounded-degree graphs)

For an arbitrary graph G, there is a quantum algorithm which uses $O(d \log m)$ copies of $|G\rangle$, and

- *For each vertex v that has degree at most d, outputs "all the neighbours of v and that v has degree at most d".*
- *For each vertex w that has degree larger than d, the algorithm outputs "degree larger than d".*

A simple but useful lemma for parity query model

Lemma 4

Let A be the adjacency matrix of G. For any $\mathrm{s} \in \{0,1\}^n$, there is a quantum algorithm which returns $A\mathrm{s}$ and makes two parity queries.

A simple but useful lemma for parity query model

Lemma 4

Let A be the adjacency matrix of G. For any $\mathbf{s} \in \{0,1\}^n$, there is a quantum algorithm which returns $A\mathbf{s}$ and makes two parity queries.

Proof.

Recall that $f(\mathbf{x}) = \sum_{(i,j)\in E} x_i x_j = \mathbf{x}^T B\mathbf{x}$, where $A = B + B^T$. Let $g(\mathbf{x}) = f(\mathbf{x}) + f(\mathbf{x}+\mathbf{s})$. We evaluate g in superposition to produce

$$\frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{g(\mathbf{x})}|\mathbf{x}\rangle = \frac{1}{\sqrt{2^n}}(-1)^{\mathbf{s}^T B\mathbf{s}} \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{\mathbf{x}^T A\mathbf{s}}|\mathbf{x}\rangle.$$

Applying Hadamard transform returns the vector $A\mathbf{s} \mod 2$. □

This is just Bernstein-Vazirani algorithm (or Fourier sampling) applied to g.

Learn graphs of m edges using parity query

Theorem 5

There is a quantum algorithm which learns a graph with at most m edges using $O(\sqrt{m \log m})$ parity queries.

Learn graphs of m edges using parity query

Theorem 5

There is a quantum algorithm which learns a graph with at most m edges using $O(\sqrt{m \log m})$ parity queries.

Proof.

Splits the graph into low and high-degree parts.

- Learn low-degree parts by Theorem 4.
- Learn high-degree parts by Lemma 4.

□

Learn graphs of m edges using parity query

Theorem 5

There is a quantum algorithm which learns a graph with at most m edges using $O(\sqrt{m \log m})$ parity queries.

Proof.

Splits the graph into low and high-degree parts.

- Learn low-degree parts by Theorem 4.
- Learn high-degree parts by Lemma 4.

□

Thanks very much for your attention!