Quantum algorithms for learning hidden graphs

Changpeng Shao

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

Workshop on General-Purpose Quantum Computing and Information Theory (7-8, June 2022)

Quantum computers

Quantum computers are designed to solve problems faster than classical computers, e.g., integer factorization [Shor, 1994], unstructured searching [Grover, 1996], Hamiltonian simulation [Lloyd, 1996], etc.

A central goal of this field is to explore for what kinds of problems quantum computers can demonstrate advantages. One interesting area is machine learning.

- **Better time complexity.**
  e.g., quantum machine learning based on quantum linear algebra, such as principal component analysis, support vector machines, ...

- **Better query complexity.**
  e.g., quantum learning theory, i.e., learning a unknown object from queries. The quantum speedup (polynomial or exponential) in this area is usually rigorous.

A simple example: Bernstein-Vazirani algorithm

It was invented by Ethan Bernstein and Umesh Vazirani in 1992 to prove an oracle separation between complexity classes BQP and BPP. It is a subroutine of many useful quantum algorithms.

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

Result:

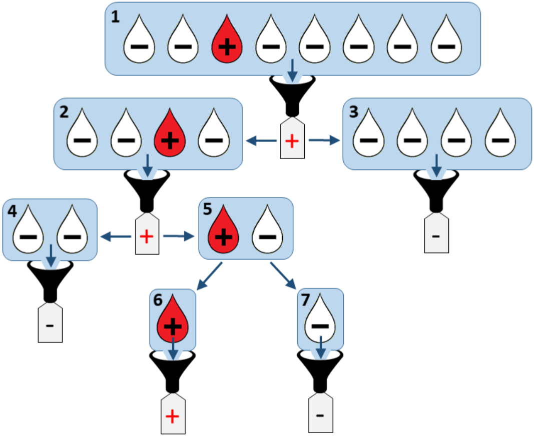
Classical vs Quantum $= n$ vs 1.

Basid idea (Fourier sampling):

$$\begin{aligned} \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \quad &\mapsto \quad \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle \\ &\mapsto \quad \frac{1}{2^n} \sum_{x,y \in \{0,1\}^n} (-1)^{f(x)+x\cdot y} |y\rangle = |a\rangle. \end{aligned}$$

A motivating example: Combinatorial group testing

First studied by Robert Dorfman in 1943 during the World War **II**.
It was used to identify syphilitic soldiers.



A motivating example: Combinatorial group testing

Problem statement: Learn Boolean function $f(x) = \vee_{j \in A} x_j$, where $\vee$ is the OR operation, $A \subseteq [n] := \{1, 2, \ldots, n\}$ is the set of affected items. For any $S \subseteq [n]$

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

Result: Assume $k = |A|$,

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

Both classical and quantum results are optimal [Belovs, 2013].

- Quadratic speedup in terms of k.
- Exponential speedup in terms of n.

This talk

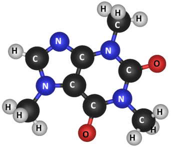
In this talk, we focus on the following learning problem, a generalization of combinatorial group testing.

Problem (Learning a hidden graph)

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

Motivated by its wide applications to 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.

Global queries:

1. **OR query**
   For any $S \subseteq V$, determines if S contains any edges.
2. **Additive query**
   For any $S \subseteq V$, returns the number of edges in S.
3. **Subset query**
   For any $S \subseteq V \times V$, determines if S contains any edges.

Queries considered in this talk

For certain problems (e.g., graph parameter estimation, number of edges, triangles, etc), global queries are exponentially efficient than local queries, e.g., [Beame, et al 2017], [Chen, Levy, Waingarten, 2020], ...

The quantum speedup for local queries is usually straightforward by Grover's algorithm.

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

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

O: upper bound, Ω: lower bound, $\Theta = O + \Omega$.

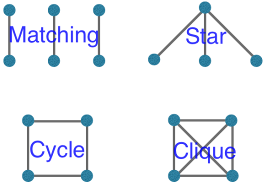
	Quantum (Our results)	Classical	
All graphs	$\Theta(n^2)$	$\Theta(n^2)$	No speedup
m edges	$O(m \log(m \log n))$ $\Omega(m)$	$O(m \log n)$[1] $\Omega(m \log \frac{n^2}{m})$	Mild speedup when $m \ll n$
Matching	$O(m^{3/4}),\ \Omega(m^{1/2})$	$\Omega(m \log \frac{n}{m})$[2]	Polynomial speedup
Cycle	$O(m^{3/4}),\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

[1] Angluin, Chen, 2008

[2] Alon, Beigel, Kasif, Rudich, Sudakov, 2004

[3] Grebinski, Kucherov, 1997

[4] Bouvel, Grebinski, Kucherov, 2005

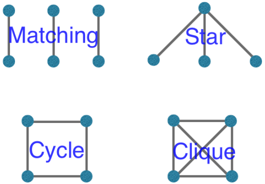


Parity query model

In the table, $m = \#$ edges, $n = \#$ vertices.
O: upper bound, Ω: lower bound, $\Theta = O + \Omega$.

	Quantum (Our results)	Classical	
All graphs	$\Theta(n)$	$\Theta(n^2)$	Quadratic
m edges	$O(\sqrt{m \log m})$	$\Omega(m \log \frac{n^2}{m})$ [1]	speedup
Matching	$O(\log m)$	$\Omega(m \log \frac{n}{m})$ [2]	
Cycle	$O(\log m)$	$\Omega(m \log \frac{n}{m})$ [3]	Exponential
Star	$O(1)$	$\Omega(m \log \frac{n}{m})$ [3]	speedup
k-vertex clique	$O(1)$	$\Omega(k \log \frac{n}{k})$ [3]	

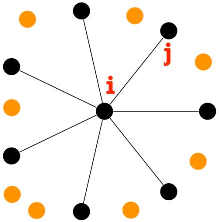
[1] Bshouty, Mazzawi, 2011
[2] Grebinski, Kucherov, 2000
[3] Bouvel, Grebinski, Kucherov, 2005



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 $i \in [n] = \{1, 2, \ldots, n\}$, the edges are $(i, j), j \in A \subseteq [n]$.

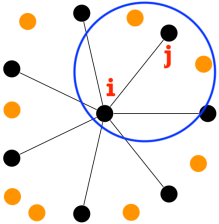


- It is equivalent to learn a Boolean function $f = x_i \wedge (\vee_{j \in A} x_j)$.
  For any $X \subseteq [n]$,

$$\begin{aligned} \mathrm{OR}(X) = 1 \quad &\Leftrightarrow \quad i, j \in X \text{ for some } j \in A \\ &\Leftrightarrow \quad f(X) = 1 \end{aligned}$$

Example 1: Learning stars by OR query

- Suppose the center is $i \in [n] = \{1, 2, \ldots, n\}$, the edges are $(i, j), j \in A \subseteq [n]$.



- It is equivalent to learn a Boolean function $f = x_i \wedge (\vee_{j \in A} x_j)$. For any $X \subseteq [n]$,

$$\begin{aligned} \mathrm{OR}(X) = 1 \quad &\Leftrightarrow \quad i, j \in X \text{ for some } j \in A \\ &\Leftrightarrow \quad f(X) = 1 \end{aligned}$$

Example 1: Learning stars by OR query

- 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 amplitude of $|0, ..., 0, 1, 0, ..., 0\rangle$, where 1 is in the i-th qubit, is $1 - 2^{1-m}$, where $m = |A|$. Perform measurements, with probability $(1 - 2^{1-m})^2$ we obtain the center i.
- When we know i, we set $x_i = 1$ in f. Now it remains to learn $f' = \vee_{j \in A} x_j$. This is a group testing problem.
- Overall cost: $\Theta(\sqrt{|A|})$.

Example 2: Learning stars by Parity query

- Suppose the center is i, the edges are $(i, j), j \in A$.
- Now it is equivalent to learn

$$f(x) = \sum_{(i,j) \in E} x_i x_j \mod 2 = x_i \sum_{j \in A} x_j \mod 2.$$

- By Fourier sampling, we obtain

$$\frac{1}{\sqrt{2}} |0, \dots, 0\rangle |+\rangle |0, \dots, 0\rangle$$
$$+ \frac{1}{\sqrt{2}} |[1 \in A], \dots, [i-1 \in A]\rangle |-\rangle |[i+1 \in A], \dots, [n \in A]\rangle,$$

where

- $|\pm\rangle = (|0\rangle \pm |1\rangle)/\sqrt{2}$ is the Hadamard basis.
- $[j \in A] = 1$ if $j \in A$ and 0 otherwise.

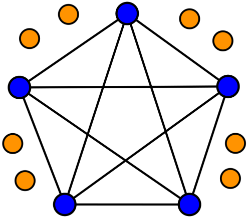
Overall cost: $O(1)$

Example 3: Learning k-clique by OR query

Basic idea:

1. We find a vertex $\mathbf{v}$ in the clique
2. Reduce the problem to a group testing problem. (Easy)

   Query with subsets of the vertices that includes $\mathbf{v}$. Such a query returns 1 iff the subset includes another vertex of the clique.

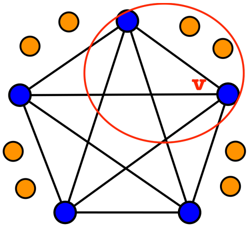


Example 3: Learning k-clique by OR query

Basic idea:

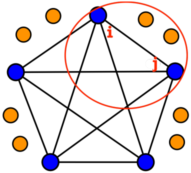
1. We find a vertex $\mathbf{v}$ in the clique
2. Reduce the problem to a group testing problem. (Easy)

   Query with subsets of the vertices that includes $\mathbf{v}$. Such a query returns 1 iff the subset includes another vertex of the clique.



Example 3: Learning k-clique by OR query

Regarding the first step, we produce a subset $S \subseteq V$ such that for any $\mathbf{v} \in V$



$$\mathrm{Prob}[\mathbf{v} \in S] = \frac{1}{k}.$$

Then with probability

$$\binom{k}{2} k^{-2}(1-1/k)^{k-2} \approx 1/2e \approx 0.18,$$

S contains exactly 2 vertices of the clique.

To learn the 2 vertices in S, it is equivalent to learn $f = x_i x_j$ for unknown i, j. By Fourier sampling, we can identify i or j.

$$\frac{1}{2}\Big(|0\rangle_i|0\rangle_j + |0\rangle_i|1\rangle_j + |1\rangle_i|0\rangle_j + |1\rangle_i|1\rangle_j\Big) \otimes |0\cdots 0\rangle_{\overline{ij}}$$

Overall cost: $\Theta(\sqrt{k})$

Example 4: Learning k-clique by Parity query

Recall that the oracle is

$$f(x) = \sum_{(i,j)\in E} x_i x_j = x^T B x,$$

where the sum is taken mod 2, B is the low-triangular part of the adjacency matrix of the graph.

Proposition

Let A be the adjacency matrix. For any $v \in \{0,1\}^n$, we can compute Av by making 2 queries of f.

Proof.

Apply the Bernstein-Vazirani algorithm to $f(x) + f(x+v)$. $\square$

Example 4: Learning k-clique by Parity query

To learn a k-clique, we choose $v = (1, \ldots, 1)^T$. For example,

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}, \quad Av = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}.$$

This idea works when k is even. If k is odd, we first find a vertex, then delete it.

Overall cost: $O(1)$

Example 5: Learning graphs from graph states

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.

A convenient way of representing entangled states, useful in quantum error-correcting codes and measurement based quantum computing models.

Bell sampling

Pauli matrices (up to applying $-i$ to σ_{11}):

$$\sigma_{00} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \sigma_{01} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_{10} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \sigma_{11} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix},$$

Bell basis:

$$|\sigma_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \quad |\sigma_{01}\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle),$$
$$|\sigma_{10}\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle), \quad |\sigma_{11}\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle).$$

Bell sampling: measure the state in the Bell basis.

Bell sampling

Proposition (Montanaro, arXiv:1707.04012)

Let $|\psi\rangle$ be a state of n qubits. Bell sampling applied to $|\psi\rangle^{\otimes 2}$ returns outcome $\sigma_s \in \{\sigma_{00}, \sigma_{01}, \sigma_{10}, \sigma_{11}\}^{\otimes n}$ with probability

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

where $|\psi^*\rangle$ is the complex conjugate of $|\psi\rangle$.

For graph state $|G^*\rangle = |G\rangle$, and $|\langle G|\sigma_s|G\rangle| = 1$ if σ_s is a stabilizer of G, otherwise $|\langle G|\sigma_s|G\rangle| = 0$.

Bell sampling for graph state

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

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

View S as a bit sting $\mathbf{s}$, then it corresponds to $A\mathbf{s}$, where A is the adjacency matrix.

Bell sampling returns a uniformly random bit string $\mathbf{s}$ and $A\mathbf{s}$.

Learning graphs from graph states

Proposition

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 $k = O(\log |\mathcal{F}|)$ Bell samples, then we obtain Boolean matrices B and AB, from which we can uniquely determine A. □

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 $|G\rangle$.

Thank you very much!