Symmetries, graph properties, and quantum speedups

Andrew Childs
University of Maryland

UMIACS
University of Maryland
Institute for Advanced
Computer Studies

JOINT CENTER FOR
QUANTUM INFORMATION
AND COMPUTER SCIENCE

Shalev Ben-David
University of Waterloo

András Gilyén
Caltech/Berkeley

William Kretschmer
UT Austin

Supartha Podder
University of Ottawa

Daochen Wang
University of Maryland

arXiv:2006.12760 / FOCS 2020

The power of quantum computers

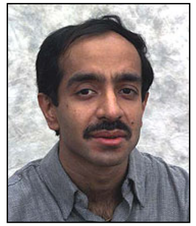
Using carefully designed interference between different computational paths, quantum computers can solve some problems dramatically faster than classical computers.

Some problems admit exponential quantum speedup.

Period finding, factoring, discrete log, quantum simulation, quantum linear algebra, Jones polynomial approximation, counting points on curves, graph connectivity with cut queries, …

Some problems only admit polynomial quantum speedup.

Unstructured search, formula evaluation, collision finding, network flows, finding subgraphs, minor-closed graph properties, group commutativity, convex optimization, …



Why? Quantum computers need a lot of structure to get significant speedup.

Query complexity

The model of *query complexity* provides a useful way of exploring the relative power of classical and quantum computers.

Main idea: Input string is described by a black box that can be queried to learn any given character. How many queries are needed to learn some Boolean property of the input?

- *Deterministic* query complexity, D: how many queries are needed for a deterministic classical algorithm to produce the correct answer?
- *Randomized* query complexity, R: how many queries are needed for a randomized classical algorithm to produce the correct answer with probability at least 2/3?
- *Quantum* query complexity, Q: how many queries are needed for a quantum algorithm to produce the correct answer with probability at least 2/3?

Example: Input describes an n-bit string. Compute the logical OR of the bits.

$$D(\mathrm{OR}) = \Theta(n) \qquad R(\mathrm{OR}) = \Theta(n) \qquad Q(\mathrm{OR}) = \Theta(n^{1/2})$$

Promise problems

General quantum query problem:

Given a black-box input $x \in P \subseteq \Sigma^n$

A query to the black box reveals x_i for any specified $i \in [n] := \{1, \ldots, n\}$

Goal: compute $f(x)$, where $f : P \to \{0, 1\}$, using as few queries as possible

If $P = \Sigma^n$ then we say f is *total*. Otherwise f is *partial*, or a *promise problem*.

The promise that x comes from P can significantly affect the difficulty of the problem.

If f is total then $R(f) = O(Q(f)^6)$ [Beals, Buhrman, Cleve, Mosca, de Wolf 01]

$R(f) = O(Q(f)^4)$ [Aaronson, Ben-David, Kothari, Rao, Tal 20]

So promises are necessary for exponential quantum speedup.

Structure and speedup

But having a promise is not enough! Some promises allow significant speedup; others don't.

Example: Collision problem.

Promise on $x \in \Sigma^n$: Either 1-to-1 (all x_is distinct; then $f(x) = 0$)
or 2-to-1 (x_is come in pairs; then $f(x) = 1$)

permutation-symmetric

$R(f) = \Theta(n^{1/2}), Q(f) = \Theta(n^{1/3})$

Example: Simon's problem.

Identify the indices of $x \in \Sigma^n$ with $\log_2 n$-bit strings

Promise on $x \in \Sigma^n$: Either 1-to-1 (then $f(x) = 0$)
or there exists an $s \neq 0$ such that $x_i = x_{i \oplus s}$ (then $f(x) = 1$)

highly structured promise

$R(f) = \Theta(n^{1/2}), Q(f) = \Theta(\log n)$

Symmetry prevents speedup

More generally, we cannot have exponential quantum speedup if the problem is too symmetric.

Say a function $f : P \to \{0,1\}$ is permutation-invariant if for all $\pi \in S_n$,

$x = (x_1, \ldots, x_n) \in P \Rightarrow x \circ \pi := (x_{\pi(1)}, \ldots, x_{\pi(n)}) \in P$

and $f(x) = f(x \circ \pi)$

Theorem. If f is permutation-invariant then $R(f) = O(Q(f)^3)$.

[Chailloux 18; improves Aaronson, Ambainis 11]

What if f has less than full permutation symmetry? How much symmetry is enough to prevent exponential quantum speedup?

Graph symmetry

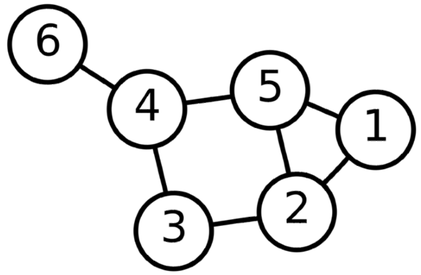
In a *graph property*, the input is a graph and the property depends only on its isomorphism class.

I.e., for any n-vertex graph $G \in P$ and $\pi \in S_n$, $\pi(G) \in P$ and $f(\pi(G)) = f(G)$

The input describes which edges are present.

A vertex permutation $\pi \in S_n$ induces an edge permutation in $S_n^{(2)} \leq S_{n^2}$, namely $\pi(u, v) = (\pi(u), \pi(v))$.

This is much less than full permutation symmetry on the edges! ($n! \ll (n^2)!$)



Graph property testing

In the setting of *property testing*, we ask if some property is satisfied or far from being satisfied.

[Goldreich, Ron 99-02]: In the adjacency-list model for bounded-degree n-vertex graphs, the randomized query complexity of testing bipartiteness and expansion is $\Theta(n^{1/2})$.

[Ambainis, Childs, Liu 11]: In the same model, the quantum query complexity of testing bipartiteness and expansion is $O(n^{1/3})$. Testing expansion requires $\Omega(n^{1/4})$ queries.

Testing bipartiteness remains wide open: Is exponential speedup possible?

Can we rule out exponential speedup for *any* graph property? [ACL 11; Montanaro, de Wolf 13]

The answer could depend on the query model (adjacency matrix vs. adjacency list).

Our results

Theorem. $R(f) = O(Q(f)^6)$ for all graph properties f in the adjacency matrix model.

More generally, $R(f) = O(Q(f)^{3k})$ for all k-uniform hypergraph properties f.

Theorem (informal). Let G be a primitive permutation group.

If G is "large" (superpolynomial base size) then any G-symmetric function has at most polynomial quantum speedup.

If G is "small" (polynomial base size) then there exists a G-symmetric function with superpolynomial quantum speedup.

Permutation groups are "large" iff they are hypergraph symmetries, so these are essentially the *only* symmetres that prevent exponential speedup.

Theorem. There is a graph property testing problem for bounded-degree graphs in the adjacency list model that can be solved with $\mathrm{poly}(\log n)$ quantum queries, but that needs $2^{\Omega(\log n)}$ classical queries.

No superpolynomial speedup for adjacency-matrix graph properties

Chailloux's proof

Suppose $f : P \subseteq \Sigma^n \to \{0,1\}$ is permutation-invariant.

Given an algorithm for computing f, if we replace the input $x \in P$ by $x \circ \pi = (x_{\pi(1)}, \ldots, x_{\pi(n)})$ for a random $\pi \in S_n$, then the algorithm still correctly computes f.

Main idea: Replace π by a random *range-r function*, $\alpha : [n] \to [n]$ with $|\alpha([n])| = r$

If a quantum algorithm distinguishes $x \circ \pi$ from $x \circ \alpha$, then it distinguishes π from α.
(If it cannot distinguish π from α then it cannot distinguish $x \circ \pi$ from $x \circ \alpha$.)

Theorem [Zhandry 15]**.** Distinguishing a random range-r function from a random permutation requires $\Omega(r^{1/3})$ quantum queries.

Taking $r = Q(f)^3$, we see that a $Q(f)$-query quantum algorithm cannot distinguish $x \circ \pi$ from $x \circ \alpha$. But a quantum algorithm on $x \circ \alpha$ can be simulated with r classical queries.

[Chailloux 18]

More general symmetries

Let $G \leq S_n$ be a permutation group. Say a function $f : P \subseteq \Sigma^n \to \{0,1\}$ is G-invariant if for all $\pi \in G$ and $x \in P$, we have $x \circ \pi \in P$ and $f(x \circ \pi) = f(x)$.

Suppose we need $\Omega(r^{1/c})$ quantum queries to distinguish a random range-r function from a random $\pi \in G$. (We say such a G is *well-shuffling*.)

Then by Chailloux's argument, $R(f) = O(Q(f)^c)$.

For graph symmetries, consider $G = S_n^{(2)}$, mapping $(u,v) \mapsto (\pi(u), \pi(v))$ for $\pi \in S_n$.

If we can distinguish a random $\pi \in S_n^{(2)}$ from a random range-r^2 function on $[n^2]$ with Q quantum queries, then we can distinguish a random $\pi \in S_n$ from a random range-r function on $[n]$ with $2Q$ quantum queries. $Q = r^{1/3} = (r^2)^{1/6}$, so $S_n^{(2)}$ is well-shuffling with $c = 6$.

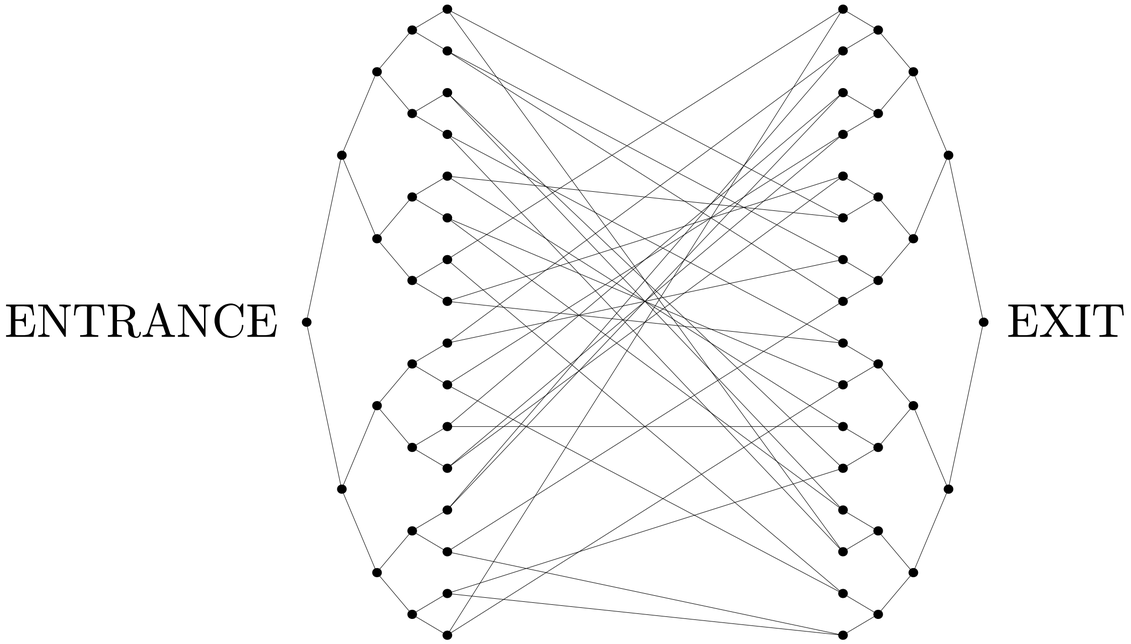
Graph symmetries have some additional constraints, but they are only "more well-shuffling".

Exponential speedup for adjacency-list graph property testing

Exponential speedup by quantum walk

Quantum analogs of random walks explore graphs in superposition.

"Welded trees" problem: given an adjacency-list black box for two binary trees joined by a random cycle, and given the name of one of the roots (the ENTRANCE), find the name of the other root (the EXIT).



Theorem. For an n-vertex welded trees graph, a quantum walk finds the EXIT in time $\mathrm{poly}(\log n)$, while a randomized classical algorithm needs time $2^{\Omega(n)}$.

This is not a graph property because the ENTRANCE is known and the property depends on how vertices are labeled.

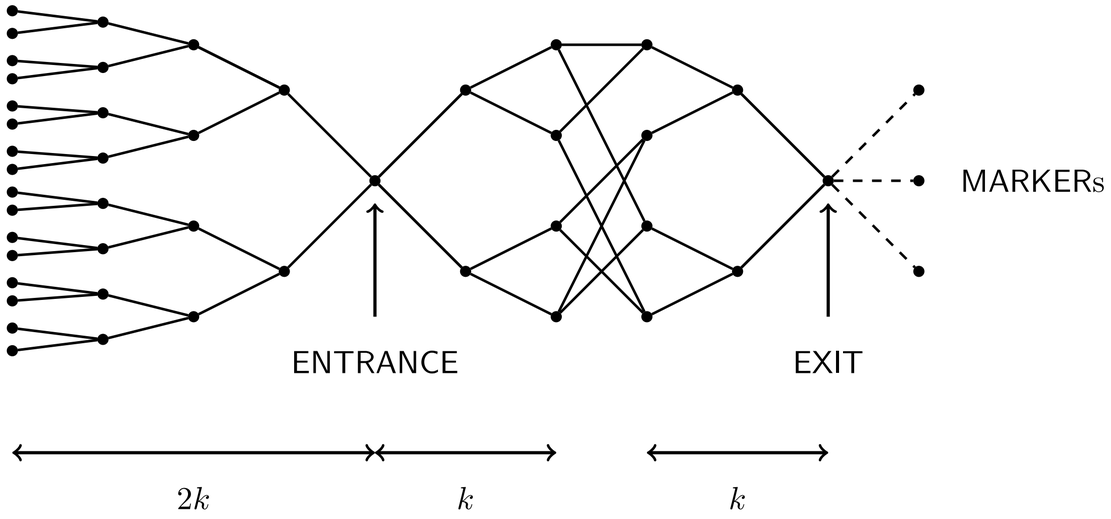
[Childs, Cleve, Deotto, Farhi, Gutmann, Spielman 03]

Graph property speedup

For "yes" instances, join three "marker" vertices to the EXIT.

Property: Is there a vertex of degree 5?

Connect a large binary tree (the "antenna") to the ENTRANCE. A random vertex is a leaf of the antenna with probability about 1/2. From such a vertex it is easy to find the ENTRANCE and then run the quantum walk to find the EXIT.



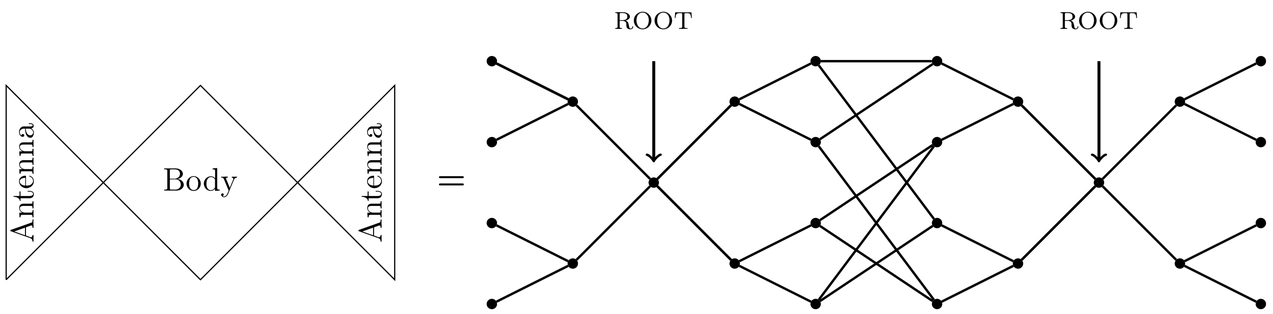
A classical algorithm still needs $2^{\Omega(n)}$ queries to solve the problem.

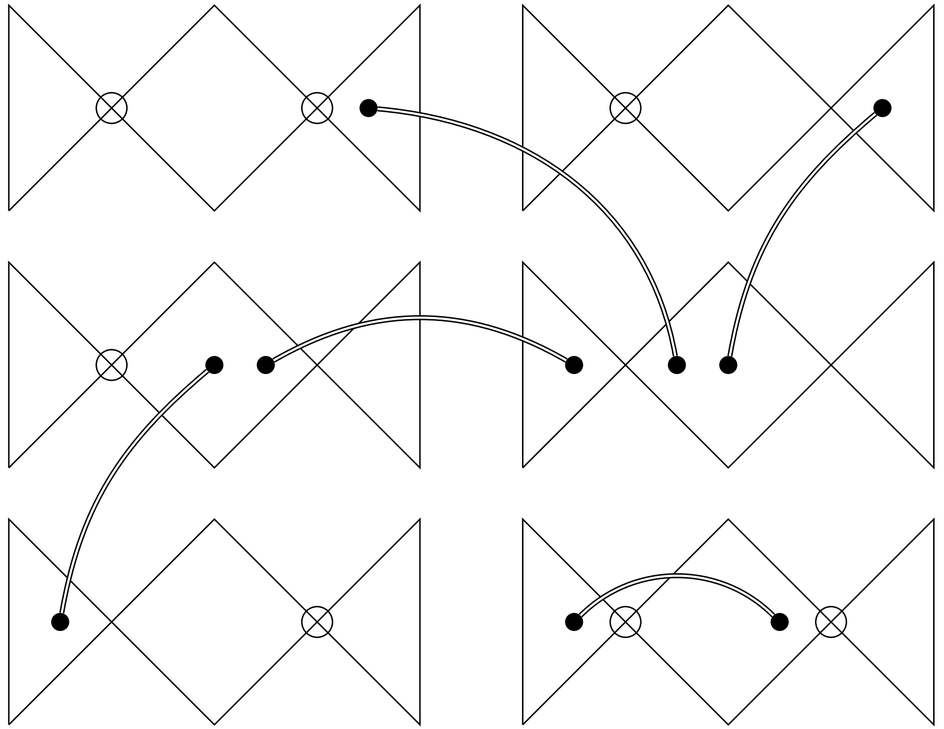
However, this is not a graph property *testing* problem: yes and no instances are close.

Graph property testing speedup

To make a property testing problem, design a graph structure that can be detected efficiently by a quantum algorithm, but that a classical algorithm has a hard time distinguishing even from very different graphs.

Main idea: Use many copies of the welded trees. Add "advice edges" that let a quantum computer test whether a vertex is in the weld. This makes it easy for a quantum computer to test the graph structure. The advice requires traversing the welded tree, so it cannot be read by a classical algorithm.





Quantum testing algorithm

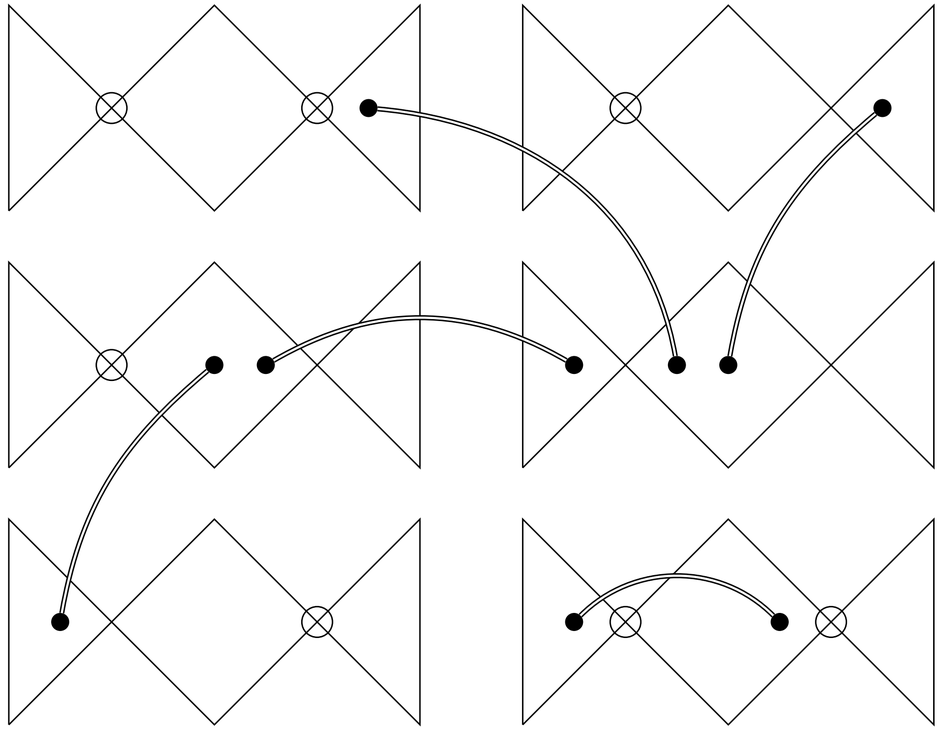
Consider "testing with advice": Given untrusted advice indicating whether each vertex is a weld or non-weld vertex, even a *classical* computer can efficiently test the property.

Testing algorithm uses non-backtracking walks to perform various consistency checks:

- binary tree structure
- welds between pairs of binary trees
- consistency of advice

A yes instance definitely passes, and we prove that a no instance is likely to be rejected.

A quantum computer can efficiently produce the advice using classical non-backtracking walks and the welded tree traversal algorithm.



Classical hardness

We show that it is hard for a classical computer to distinguish a valid yes instance from similar instances in which the trees are "self-welded":

This gives a specific set of no instances (they are typically far from the yes instances; in particular, they are typically far from bipartite).

An algorithm can only explore the graph locally. When exploring the central "body," it is hard to distinguish the graph from a large 3-regular tree, as in the lower bound for the original welded tree problem.

Open problems

What is the largest possible separation for testing (hyper)graph properties?
We know $R(f) = O(Q(f)^{3k})$ for k-uniform hypergraphs.
For $k = 1$: $R(f) = O(Q(f)^3)$; best known separation is $R(\mathrm{OR}) = \Omega(Q(\mathrm{OR})^2)$.
For $k > 1$: best known separation is $R(f) = \Omega(Q(f)^{k/2})$, using Forrelation.
Can we tighten this?

Can we fully characterize which permutation groups allow superpolynomial speedup?
(Our characterization is already close to this.)

Is there a practical graph property testing problem (in the adjacency-list model) with exponential quantum speedup? How about testing bipartiteness? Can we show that testing *monotone* graph properties cannot have exponential speedup?