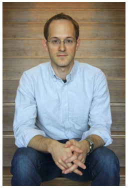
Symmetries, graph properties, and quantum speedups

Daochen Wang (Maryland)

Full version: arXiv: 2006.12760

APS March Meeting 2021

Shalev Ben-David (Waterloo)


Andrew Childs (Maryland)

András Gilyén (Caltech)

William Kretschmer (UT Austin)

Supartha Podder (Ottawa)

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 polynomial quantum speedup:
  Unstructured search, spatial search, formula evaluation, element distinctness, graph connectivity, semi-definite programming, ...
- Other problems admit super-polynomial quantum speedup:
  Period finding, factoring, discrete log, Pell's equation, quantum simulation, quantum linear algebra, quantum differential equations, ...

Why? We address this question through the lens of symmetry.

Query complexity measures quantum speedup

Let $f : \mathcal{D} \subset \{0,1\}^n \to \{0,1\}$ be a known function.

- How many positions of input $x \in \mathcal{D}$ do you need to query to compute $f(x)$ with high probability in the worst case?
- Answer denoted $R(f)$ and $Q(f)$ in the classical and quantum cases respectively. Quantumly, can query x in superposition.
- We want to know when $R(f) = Q(f)^{\omega(1)}$ (large speedup) and when $R(f) = Q(f)^{O(1)}$ (small speedup).
- Interesting facts:
  1. Small Grover speedup: $f = \mathrm{OR}$ with $\mathcal{D} \coloneqq \{0,1\}^n$ has $R(\mathrm{OR}) = \Theta(n)$ and $Q(\mathrm{OR}) = \Theta(\sqrt{n})$.
  2. $\mathcal{D}$ is very important! For example, $R(\mathrm{OR}) = Q(\mathrm{OR}) = 0$ if $\mathcal{D} = \{0,1\}^n - \{0^n\}$. In fact, for *any* f, when $\mathcal{D} = \{0,1\}^n$, there can only be small speedups[1].
  3. Large speedups exist. For example, Simon (1997) exhibited an f with $R(f) = \Theta(\sqrt{n})$ and $Q(f) = \Theta(\log(n))$.

[1] Beals, Buhrman, Cleve, Mosca, and de Wolf (2001); Aaronson, Ben-David, Kothari, and Tal (2020).

Characterization of quantum speedups for symmetric functions: “must be small for adjacency matrix hypergraph-based symmetries, else can be large”

Symmetric functions

Definition

Let $f : \mathcal{D} \subset \{0,1\}^n \to \{0,1\}$ be a function. f is symmetric under a permutation group G on $\{1, \dots, n\}$ if, for all $\pi \in G$, we have:

1. $x = (x_1, \dots, x_n) \in \mathcal{D} \implies x \circ \pi := (x_{\pi(1)}, \dots, x_{\pi(n)}) \in \mathcal{D}$.
2. $f(x) = f(x \circ \pi)$ for all $x \in \mathcal{D}$.

Examples:

- $f = \mathrm{OR} : \{000, 100, 010, 001\} \subset \{0,1\}^3 \to \{0,1\}$ is symmetric under $G = S_3$ (all permutations of $\{1, 2, 3\}$).
- $f =$ a graph property in the adjacency matrix model is symmetric under $G =$ graph isomorphisms.

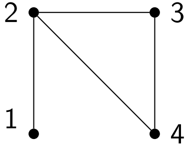
Adjacency matrix model of graphs

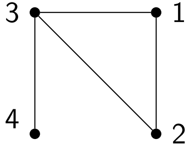
In the adjacency matrix model, a (simple) graph on vertex set $[n] := \{1, \ldots, n\}$ is modelled by a $m := \binom{n}{2}$-bit string

For example, let $n = 4$, so $m = 6$, under the index-edge identification:

$$\begin{aligned} &1 \leftrightarrow \{1,2\},\ 2 \leftrightarrow \{1,3\},\ 3 \leftrightarrow \{1,4\}, \\ &4 \leftrightarrow \{2,3\},\ 5 \leftrightarrow \{2,4\},\ 6 \leftrightarrow \{3,4\}, \end{aligned} \tag{1}$$

the left graph is 100111 and the right graph is 110101.



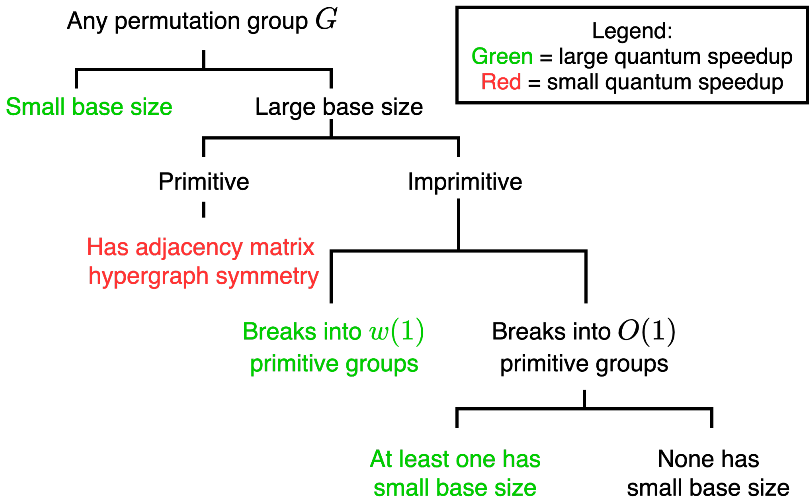


A graph property in the adjacency matrix model is a function on such bitstrings that is invariant under graph isomorphism.

Near-complete characterization theorem

Prior art[2]: small quantum speedup for f symmetric under $G = S_n$.

Our theorem:


Any permutation group G
Legend:
Green = large quantum speedup
Red = small quantum speedup
Small base size
Large base size
Primitive
Imprimitive
Has adjacency matrix hypergraph symmetry
Breaks into $w(1)$ primitive groups
Breaks into $O(1)$ primitive groups
At least one has small base size
None has small base size

[2] Aaronson and Ambainis (2009); Chailloux (2018).

Chailloux's proof (2018)

Suppose $f : \mathcal{D} \subset \{0,1\}^n \to \{0,1\}$ is invariant under S_n.

Given an algorithm for computing f, if we replace the input $x \in \mathcal{D}$ 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 (2015)]. Distinguishing a random range-r function from a random permutation in S_n 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.

Adjacency matrix graph symmetries

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 $G = S_n^{(2)}$ on $[n^2]$, consisting of mappings $(u, v) \in [n^2] \mapsto (\pi(u), \pi(v))$ 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. So $2Q = \Omega(r^{1/3}) = \Omega((r^2)^{1/6})$, so $S_n^{(2)}$ is well-shuffling with $c = 6$.

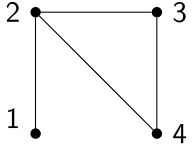
Graph symmetries have some additional constraints, but they are only "more well-shuffling".

There exists an exponential quantum speedup for graph property testing in the adjacency list model

Adjacency list model of graphs

In the adjacency list model, a (simple) graph of bounded degree d on vertex set $[n]$ is modelled by a $n \times d$ matrix – which can then be collapsed into a length-(nd) string.

For example, the graph (seen before):

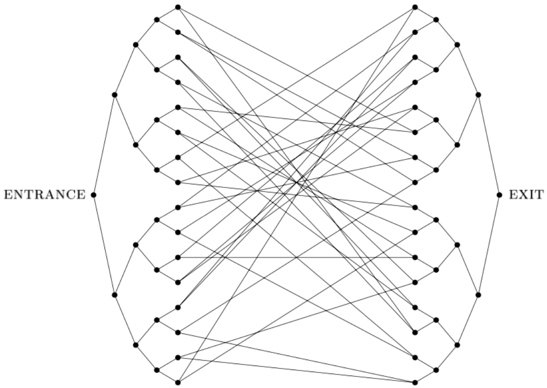


with $n = 4, d = 3$ can be modelled by

$$x = \begin{bmatrix} 2 & * & * \\ 1 & 3 & 4 \\ 4 & 2 & * \\ 2 & 3 & * \end{bmatrix} \tag{2}$$

The glued trees problem

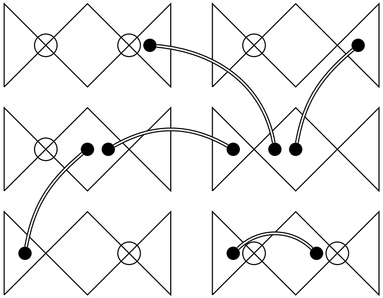
Given access to the adjacency list of a glued trees graph and the label of ENTRANCE, a quantum algorithm can find the label of EXIT exponentially faster than any classical algorithm[3].



[3]Childs, Cleve, Deotto, Farhi, Gutmann, and Spielman (2003).

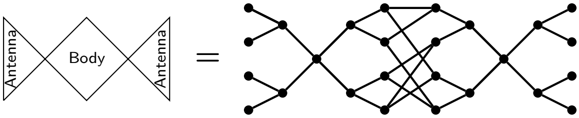
Use glued trees to construct a property testing problem with exponential quantum speedup

The graph property:



1. Can *classically* test the *entire* glued-trees if we mark the leaves of the two trees that are glued.
2. Mark the leaves in a way that can only be read efficiently by a quantum computer but not a classical computer – use further copies of the glued-trees problem.

where



In particular: quantum speedups of computing graph properties depend significantly on the input model!

Adjacency list: an exponential quantum speedup exists even for graph property testing.

Adjacency matrix: there can be at most polynomial quantum speedup, $R(f) = O(Q(f)^6)$.

These results resolve an open question of Ambainis, Childs, and Liu (2010) and Montanaro and de Wolf (2013).

Outlook

Thank you for your attention! Here are a few of the interesting questions remaining from our work:

1. We showed $R(f) = O(Q(f)^{3p})$ for p-uniform hypergraph properties f in the adjacency matrix model as part of our characterization theorem. How tight is this?
2. Can we complete our characterization theorem?
3. Is there a *useful* graph property testing problem in the adjacency list model with super-polynomial quantum speedup?