Symmetries, graph properties, and quantum speedups

Daochen Wang
University of Maryland

Joint work with Shalev Ben-David, Andrew M. Childs, András Gilyén, William Kretschmer, and Supartha Podder

arXiv: 2006.12760

Microsoft Research
6th July 2020

Outline

Introduction

Query complexity (1/4)

The first problem. Let $f : \{0,1\}^n \to \{0,1\}$ be known in advance. Given *unknown* input $x \in \{0,1\}^n$ to f. How many bits of x do you need to deterministically read (aka query) to compute f?

Examples:

1. $f = \mathrm{OR}$, i.e. $f(x) = 1$ if and only if at least one bit of x is a 1.
2. $f(x) = x_1$.
3. $f(x) = (x_1 \wedge x_2 \wedge x_3) \vee x_3$.

The answer is known as the deterministic query complexity of f, denoted $D(f)$. If we can use random-ness and only require the output to be correct with probability at least $2/3$, then the answer is known as the randomized query complexity of f, denoted $R(f)$.

Query complexity (2/4)

If we can use *quantum-ness* and only require the output to be correct with probability at least 2/3, then the answer is known as the quantum query complexity of f, denoted $Q(f)$.

More precisely, quantum-ness means we can do quantum computations and have access to the *quantum oracle*

$$\begin{aligned} O_x : \mathbb{C}^n \otimes \mathbb{C}^2 &\to \mathbb{C}^n \otimes \mathbb{C}^2 \\ |i\rangle \otimes |b\rangle &\mapsto |i\rangle \otimes |b \oplus x_i\rangle \,. \end{aligned} \tag{1}$$

This means we can query the bits of x in *superposition*.

Fact: $Q(f) \leq R(f) \leq D(f)$.

Query complexity (3/4)

More generally, can consider $f : \mathcal{D} \subset \Sigma^n \to \{0,1\}$. Σ is known as the input alphabet, previously $\Sigma = \{0,1\}$. The domain $\mathcal{D}$ is known as the promise on the input $x \in \Sigma^n$. When $\mathcal{D} = \Sigma^n$, f is said to be *total*, else it is said to be *partial*. The query complexity of f can depend *significantly* on the promise.

Examples:

1. $f = \mathrm{OR}$ and $\Sigma = \{0,1\}$, but now $\mathcal{D} = \{0^n\}^c$, i.e. promised input is *not* 0^n, the all-zeros bitstring.
2. When f is total and $\Sigma = \{0,1\}$, then[1] $R(f) \leq D(f) = O(Q(f)^4)$. In particular, no exponential speedups.

(It may help to think of $x = O(y)$ as $x \leq y$ and $x = \Omega(y)$ as $x \geq y$ because we don't care about constants.)

[1] Aaronson, Ben-David, Kothari, and Tal (2020).

Query complexity (4/4)

Still consider $f : \mathcal{D} \subset \Sigma^n \to \{0,1\}$. Input $x \in \mathcal{D} \subset \Sigma^n$, x can be viewed as a function from $[n]$ to Σ.

Collision problem. $\Sigma = [n] := \{1, 2, \dots, n\}$. Promised that x is either 1-to-1 ($f = 0$) or ($k > 1$)-to-1 ($f = 1$).

$Q(f) = \Theta((n/k)^{1/3})$; $R(f) = \Theta((n/k)^{1/2})$. Polynomial speedup.

Simon's problem. $\Sigma = [n]$, where $n = 2^k$. View the n indices of x as labelled by $\{0,1\}^k$. Promised that either x is 1-to-1 ($f = 0$) or there exists an $a \neq 0^k$ such that $x_i = x_{i \oplus a}$ for all i ($f = 1$).

$Q(f) = \Theta(k = \log_2 n)$; $R(f) = \Theta(\sqrt{n})$. Exponential speedup!

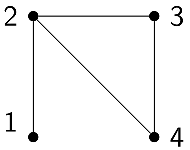
Models of graphs: adjacency matrix

In the adjacency matrix model, a (simple) graph on vertex set $[n] = \{1, \ldots, n\}$ is modelled by a $\binom{n}{2}$-bit string, where the indices are first identified with edges and the bit-value at an index indicates whether that edge is present.

For example, under the following index-edge identification:

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

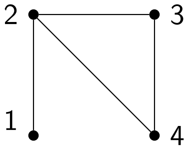
the graph below with $n = 4$ is modelled by $x = 100111$.



Models of graphs: adjacency list

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 (same as before):



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

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

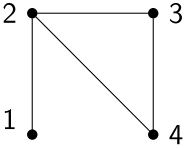
among other possibilities.

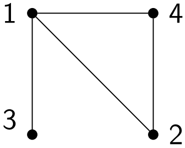
Graph properties

A graph property f is a function from a set of graphs (specified either in the adjacency matrix or list model) to $\{0, 1\}$ that is invariant under graph isomorphisms, i.e. vertex relabellings.

Examples:

1. Having a triangle or not is a graph property.
2. f must evaluate to the same value on the following two isomorphic graphs. Note that the graphs are not the *same*, e.g. in the adjacency matrix model, the left one is $x = 100111$ but the right one is $x = 111010$ (under the same index-edge identification as before).





Symmetries of graphs in adjacency matrix model

Symmetric functions

Definition

A permutation group G of $[n]$ is a set of permutations of $[n]$ that forms a group. To say a function $f : \mathcal{D} \subset \Sigma^n \to \{0,1\}$ is symmetric under G means, for all $\pi \in G$:

1. If $x \in \mathcal{D}$ then $x \circ \pi \in \mathcal{D}$, where $x \circ \pi \in \Sigma^n$ is defined by $(x \circ \pi)_i = x_{\pi(i)}$.
2. $f(x) = f(x \circ \pi)$ for all $x \in \mathcal{D}$. (Note that the RHS makes sense by the first condition.)

Main example. f is a graph property, $\Sigma = \{0,1\}$, and G are graph symmetries denoted S_n^2, i.e. the set of permutations of $[n = \binom{m}{2}]$ *induced* by the S_m permutations of vertex set $[m]$. More generally, f is a p-uniform hypergraph property and $G = S_n^p$. (Fix $p = 2$ if hypergraphs are unfamiliar.)

Permutation groups and small-range strings

A permutation group G of $[n]$ can be identified with a set of length-n strings in a natural way. For example, the permutation of $[3]$ that maps

$$1 \mapsto 3, \quad 2 \mapsto 1, \quad 3 \mapsto 2 \tag{4}$$

is identified with the 3-bit string "312".

Let $1 < r < n$ be an integer. Consider another subset of length-n strings $D_{n,r}$ defined by having at most r distinct entries in $[n]$. For example:

$$\begin{aligned} D_{3,2} = \{&111, 222, 333, \\ &112, 121, 211, 221, 212, 122, \\ &113, 131, 311, 331, 313, 133, \\ &223, 232, 322, 332, 323, 233\}. \end{aligned} \tag{5}$$

$D_{n,r}$ is known as a set of small-range strings (with range r). Note that $D_{n,r}$ is disjoint from G, i.e. $D_{n,r} \cap G = \emptyset$.

Well-shuffling permutation groups

We say a permutation group is well-shuffling if it is hard for a quantum computer to distinguish it from small-range strings.

More precisely:

Definition

Let G be a permutation group of $[n]$. We say that G is well-shuffling with power $a > 0$ if $\mathrm{cost}(G, r) := Q(f_{G,r}) = \Omega(r^{1/a})$, where we define

$$f_{G,r} : G \dot{\cup} D_{n,r} \to \{0, 1\}$$
$$x \mapsto \begin{cases} 0 & \text{if } x \in G \\ 1 & \text{if } x \in D_{n,r} \end{cases} . \quad (6)$$

Well-shuffling implies R and Q are polynomially close

Theorem

Let $f : \mathcal{D} \subset \Sigma^n \to \{0,1\}$ be symmetric under G. Then, there exists a $c > 0$ such that: if $Q(f) \le \mathrm{cost}(G,r)/c$ then $R(f) \le r$. Hence: if G is well-shuffling with power a then $R(f) = O(Q(f)^a)$.

Proof sketch[2].

1. Let Q be a quantum algorithm computing f using $Q(f)$ queries to O_x, where $x \in \mathcal{D}$ is the input.
2. Replacing all O_x by $O_{x \circ \pi}$ where $\pi \in G$ doesn't change the output much. Because f is symmetric under G.
3. Then replacing $O_{x \circ \pi}$ by $O_{x \circ \alpha}$ doesn't change the output much, where $\alpha \in D_{n,r}$ and $x \circ \alpha$ is the length-n string with entries $(x \circ \alpha)_i = x_{\alpha_i}$. Because $Q(f) \le \mathrm{cost}(G,r)/c$.
4. The last quantum circuit queries at most r entries of x, so can simulate by a randomized algorithm using at most r queries.

□

[2] Chailloux (2018).

Hypergraph symmetries are well-shuffling (1/2)

$(p = 1)$-uniform hypergraph symmetries are exactly those in the full permutation group $G = S_n$ of $[n]$.

Theorem

S_n is well-shuffling with power 3.

Proof.

1. Unpack the statement: suppose we have a quantum algorithm Q that distinguishes between length-n strings x with at most r distinct entries from ones that are 1-to-1, then Q must use $\Omega(r^{1/3})$ queries to O_x.
2. But we can run Q to distinguish between length-n strings that are (n/r)-to-1 from ones that are 1-to-1, that is, solve the collision problem. So Q must use $\Omega(r^{1/3})$ queries by the lower bound for the collision problem.

□

Hypergraph symmetries are well-shuffling (2/2)

p-uniform hypergraph symmetries form a permutation group $G = S_n^p$ of $[\binom{n}{p}]$ induced by the permutation group S_n of $[n]$.

Theorem

S_n^p is well-shuffling with power $3p$.

Proof sketch.

1. Instead of S_n^p, first prove the same statement for permutation group $S_n^{(p)}$ of $[n^p] = [n]^p$ that consists of permutations $\bar{\pi}$ that map $(i_1, i_2, \ldots, i_p) \in [n]^p$ to $(\pi(i_1), \pi(i_2), \ldots, \pi(i_p))$.
2. If can distinguish $S_n^{(p)}$ from $D_{n^p, s:=r^p}$ using Q queries, then can distinguish S_n from $D_{n,r}$ using $O(pQ)$ queries, which is at least $\Omega(r^{1/3} = s^{1/(3p)})$. So $Q = \Omega(s^{1/(3p)}/p)$. So $S_n^{(p)}$ is well-shuffling with power $3p$.
3. Not hard to see that S_n^p is "more well-shuffling" than $S_n^{(p)}$, which gives the Theorem.

□

Computing hypergraph properties admits at most a polynomial quantum speedup

We have shown:

Theorem

Let $f : \mathcal{D} \subset \Sigma^n \to \{0,1\}$ be symmetric under G. Then, there exists a $c > 0$ such that: if $Q(f) \leq \mathrm{cost}(G,r)/c$ then $R(f) \leq r$. If G is well-shuffling with power a, then $R(f) = O(Q(f)^a)$; and

Theorem

S_n^p is well-shuffling with power $3p$.

But a p-uniform hypergraph property is symmetric under $G = S_n^p$, which is well-shuffling with power $3p$. Hence:

Corollary

$R(f) = O(Q(f)^{3p})$ for any p-uniform hypergraph property f.

Symmetries of primitive permutation groups

Base of permutation groups and quantum speedups (1/3)

Definition

A base of a permutation group G of $[n]$ is a set $S \subset [n]$ such that if $h \in G$ and $h(x) = x$ for all $x \in S$ then h is the identity element in G. The base size $b(G)$ of G is the minimal size of a base.

Examples:

1. S_3 of $[3]$ has base size 2; a base is $\{1, 2\}$;
   S_n of $[n]$ has base size $n - 1$; a base is $\{1, 2, \ldots, n-1\}$.
2. $\mathrm{GL}_n(\mathbb{F}_2)$, invertible $n \times n$ matrices over $\mathbb{F}_2$, of $\mathbb{F}_2^n$ has base size n; a base is $\{(1, 0, \ldots, 0), \ldots, (0, 0, \ldots, 1)\}$ (standard basis of $\mathbb{F}_2^n$). Note that the base size is very small in the sense that it equals $\log_2(|\mathbb{F}_2^n| = 2^n)$.
3. If $h, k \in G$ agree on a base, then hk^{-1} fixes that base, so $h = k$ by definition. So if you know how h behaves on a base, you can identify h.

Base of permutation groups and quantum speedups (2/3)

Theorem

Let G be a permutation group of $[n]$, and let $f : \mathcal{D} \subset \Sigma^n \to \{0,1\}$. Then, there exists a partial Boolean function h that is symmetric under G such that $Q(h) \leq Q(f) + b(G)$ and $R(h) \geq R(f)$.

Proof sketch.

Example: $n = 2$, $\mathcal{D} = \{(a,a),(b,a)\} \subset \Sigma^n = \{a,b\}^2$ and $G = S_2$. Construct the set $\mathcal{D}_G$ of "G-permutations of $\mathcal{D}$":

$$\mathcal{D}_G := \{[(a,1),(a,2)],\ [(a,2),(a,1)],\ [(b,1),(a,2)],\ [(a,2),(b,1)]\} \\ \subset (\Sigma \times [n])^n = \{(a,1),\ (a,2),\ (b,1),\ (b,2)\}^2 \tag{7}$$

and let h be "the same as" f. Then $h : \mathcal{D}_G \subset (\Sigma \times [n])^n \to \{0,1\}$ is *by definition* symmetric under G. $Q(h) \leq Q(f) + b(G)$: query the indices in the base to identify the G-permutation, then reverse this permutation, and use algorithm for computing f to compute h. $R(h) \geq R(f)$: clear as h is harder to compute than f. □

Base of permutation groups and quantum speedups (3/3)

Theorem

Let G be a permutation group of $[n]$, and let $f : \mathcal{D} \subset \Sigma^n \to \{0,1\}$. Then, there exists a partial Boolean function h that is symmetric under G such that $Q(h) \le Q(f) + b(G)$ and $R(h) \ge R(f)$.

Consequence. If G has base size $b(G) = O(n^{o(1)})$, then we can construct a h that is symmetric under G *and* possesses a super-polynomial speedup as follows.

In the Theorem above take f to be the function in Simon's problem, then $Q(f) = O(\log n)$, but $R(f) = \Omega(\sqrt{n})$. Therefore

$$\begin{aligned} &Q(h) \le Q(f) + b(G) = O(\log n) + O(n^{o(1)}) = O(n^{o(1)}), \\ &R(h) \ge R(f) = \Omega(\sqrt{n}). \end{aligned} \tag{8}$$

This represents a super-polynomial speedup by definition.

Primitive permutation groups

Primitive permutation groups are special types of transitive permutation groups that are the "building-blocks" of all permutation groups.

Theorem (Liebeck, 1984)

Let G be a primitive permutation group of $[n]$. Then one of the following cases hold:

1. $n = \binom{m}{p}^{\ell}$ *and G contains permutations of* $[n] = [\binom{m}{p}]^{\ell}$ *that permutes each of the ℓ-entries according to* $A_m^p \subset S_m^p$ *(most p-uniform hypergraph symmetries).*
2. $b(G) < 9\log_2(n)$.

In Case 2, we can get an exponential quantum speedup via Theorem on last slide. In Case 1, we can get at most a $3\ell p$-power quantum speedup, which is polynomial for *constant* ℓ, p. The converse can be proved via Theorem on last slide: if ℓ, p are not both constant, we can get a super-polynomial quantum speedup.

Adjacency list model

Brief overview (1/2)

Main idea: upgrade the glued-trees problem[3], which has an exponential quantum speedup in the adjacency list model, to a property-testing problem.

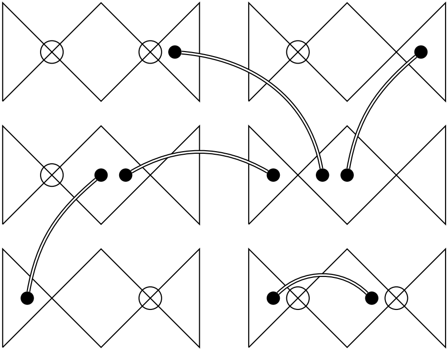
Execution:

1. can *classically* test the *entire* glued-trees structure 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.

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

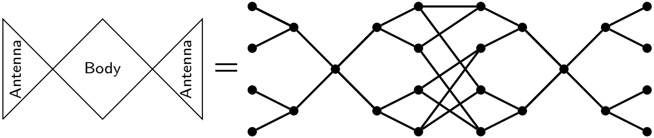
Brief overview (2/2)

The graph property (i.e. yes-instances):



Six "candy" (sub)graphs and five of the many "advice edges" (indicated by double lines) that connect each body vertex to a distinct antenna vertex. The circles in the figure represent self-loops at the roots of the candy graphs, which provide advice about whether a body vertex is a leaf or non-leaf. Even parity of circles indicates non-leaf, odd parity indicates leaf.

where



Open problems

Open problems

Thank you for your attention! Here are some of our open problems:

1. We showed that $R(f) = O(Q(f)^{3p})$ for computing p-uniform hypergraph properties f in the adjacency matrix model, but what is the largest possible separation? That is, what is the largest k for which there exists such an f with $R(f) = \Omega(Q(f)^k)$? Know $k \geq p$. Open *even for* $p = 1$.
2. Can we get a complete characterization theorem regarding which (arbitrary) permutation groups allow super-polynomial quantum speedups and which do not? Feel close already.
3. Does there exist a graph property testing problem *of practical interest* in the adjacency list model that admits an exponential or super-polynomial quantum speedup? We also conjecture that deciding a *monotone* graph property cannot admit a super-polynomial quantum speedup.

Appendix: primitive permutation groups

Definition

A primitive permutation group G of $[n]$ is a transitive permutation group such that the only partitions $\mathcal{B} := \{B_1, \ldots, B_k\}$ of $[n]$ preserved by G, i.e. $\pi(\mathcal{B}) := \{\pi(B_i)\}_i = \mathcal{B}$ for all $\pi \in G$, are $\{G\}$ and the partition into singletons.

Example of a transitive but imprimitive permutation group.
Let $n = 4$, consider permutation group $G = \langle (12)(34), (13)(24) \rangle$ of $[4]$. G is transitive, but preserves the following partition:

$$\mathcal{B} = \{B_1 = \{1, 3\}, B_2 = \{2, 4\}\}, \tag{9}$$

so is imprimitive.