Symmetries, Graph Properties, and Quantum Speedups

Shalev Ben-David*, Andrew M. Childs†‡, András Gilyén§¶,
William Kretschmer‖, Supartha Podder**, Daochen Wang‡††
*Cheriton School of Computer Science, University of Waterloo
†Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland
‡Joint Center for Quantum Information and Computer Science, University of Maryland
§Institute for Quantum Information and Matter, California Institute of Technology
¶Simons Institute for the Theory of Computing, University of California, Berkeley
‖Department of Computer Science, University of Texas at Austin
**Department of Mathematics and Statistics, University of Ottawa
††Department of Mathematics, University of Maryland

***Abstract*—Aaronson and Ambainis (2009) and Chailloux (2018) showed that fully symmetric (partial) functions do not admit exponential quantum query speedups. This raises a natural question: how symmetric must a function be before it cannot exhibit a large quantum speedup?**

In this work, we prove that hypergraph symmetries in the adjacency matrix model allow at most a polynomial separation between randomized and quantum query complexities. We also show that, remarkably, permutation groups constructed out of these symmetries are essentially the *only* permutation groups that prevent super-polynomial quantum speedups. We prove this by fully characterizing the primitive permutation groups that allow super-polynomial quantum speedups.

In contrast, in the adjacency list model for bounded-degree graphs—where graph symmetry is manifested differently—we exhibit a property testing problem that shows an exponential quantum speedup. These results resolve open questions posed by Ambainis, Childs, and Liu (2010) and Montanaro and de Wolf (2013).

***Keywords*-quantum query complexity; graph properties; property testing**

I. Introduction

One of the most fundamental problems in the field of quantum computing is the question of when quantum algorithms substantially outperform classical ones. While polynomial quantum speedups are known in many settings, super-polynomial quantum speedups are known (or even merely conjectured) for only a few select problems. Crucially, exponential quantum speedups only occur for certain "structured" problems such as period-finding (used in Shor's factoring algorithm [1]) and Simon's problem [2], in which the input is known in advance to have a highly restricted form. In contrast, for "unstructured" problems such as black-box search or NP-complete problems, only polynomial speedups are known (and in some models, it can be formally shown that only polynomial speedups are possible).

In this work, we are interested in formalizing and characterizing the structure necessary for fast quantum algorithms. In particular, we study the *types of symmetries* a function can have while still exhibiting super-polynomial quantum speedups.[1]

A. Prior work

Despite the strong intuition in the field that structure is necessary for super-polynomial quantum speedups, only a handful of works have attempted to formalize this and characterize the required structure. All of them study the problem in the query complexity (black-box) model of quantum computation, which is a natural framework in which both period-finding and Simon's problem can be formally shown to give exponential quantum speedups (see [3] for a survey of query complexity, or [4] for a formalization of period-finding specifically).

In the query complexity model, the goal is to compute a Boolean function $f\colon \Sigma^n \to \{0,1\}$ using as few queries to the bits of the input $x \in \Sigma^n$ as possible, where Σ is some finite alphabet. Each query specifies an index $i \in [n] \coloneqq \{1,2,\ldots,n\}$ and receives the response $x_i \in \Sigma$. A query algorithm, which may depend on f but not on x, must output $f(x)$ (with bounded error in the worst case) after as few queries as possible. Quantum query algorithms are allowed to make queries in superposition, and we are interested in how much advantage this gives them over randomized classical algorithms (for formal definitions of these notions, see [3]).

Beals, Buhrman, Cleve, Mosca, and de Wolf [5] showed that all *total Boolean functions* $f\colon \Sigma^n \to \{0,1\}$ have a polynomial relationship between their classical and quantum query complexities (which we denote $\mathrm{R}(f)$ and $\mathrm{Q}(f)$, respectively). This means that super-polynomial speedups are not possible in query complexity unless we impose a promise on the input: that is, unless we define $f\colon P \to \{0,1\}$ with $P \subset \Sigma^n$, and allow an algorithm computing f to behave arbitrarily on inputs outside of the promise set P. For such promise problems (also called partial functions), provable

[1]The full version of this paper is available at arXiv:2006.12760

2575-8454/20/$31.00 ©2020 IEEE
DOI 10.1109/FOCS46700.2020.00066

exponential quantum speedups are known. This is the setting in which Simon's problem and period-finding reside.

The question, then, is what we can say about the structure necessary for a partial Boolean function f to exhibit a super-polynomial quantum speedup. Towards this end, Aaronson and Ambainis [6] showed that *symmetric* functions do not allow super-polynomial quantum speedups, even with a promise. Chailloux [7] improved this result by reducing the degree of the polynomial relationship between randomized and quantum algorithms for symmetric functions, and removing a technical requirement on the symmetry of those functions.[2]

Other work attempted to characterize the structure necessary for quantum speedups in alternative ways. Ben-David [8] showed that certain types of symmetric promises do not admit any function with a super-polynomial quantum speedup, a generalization of [5] (who showed this when the promise set is Σ^n). Aaronson and Ben-David [9] showed that small promise sets, which contain only $\mathrm{poly}(n)$ inputs out of $|\Sigma|^n$, also do not admit functions separating quantum and classical algorithms by more than a polynomial factor.

A class of problems with significant symmetry, though much less than full permutation symmetry, is the class of graph properties. For such problems, the input describes a graph, and the output depends only on the isomorphism class of that graph. Thus the vertices can be permuted arbitrarily, but such a permutation induces a structured permutation on the edges, about which queries provide information.

The setting of graph property *testing* provides a natural class of partial graph properties. Here we are promised that the input graph either has a property, or is ϵ-far from having the property, meaning that we must change at least an ϵ fraction of the edges to make the property hold. Graph property testing has been extensively studied since its introduction by Goldreich, Goldwasser, and Ron [10].

The behavior of classical graph property testers can differ substantially depending on the model in which the input graph is specified. For example, in the adjacency matrix model, Alon and Krivelevich [11] proved that bipartiteness can be tested in $\tilde{O}(1/\epsilon^2)$ queries, which is surprisingly independent of the size of the input graph. In contrast, in the adjacency list model for bounded-degree graphs, Goldreich and Ron [12] proved that $\Omega(\sqrt{n})$ queries are needed to test bipartiteness of n-vertex graphs.

Quantum algorithms for testing properties of bounded-degree graphs in the adjacency list model were studied by Ambainis, Childs, and Liu [13]. They gave upper bounds of $\tilde{O}(n^{1/3})$ for testing bipartiteness and expansion, demonstrating polynomial quantum speedups. Furthermore, they showed that at least $\Omega(n^{1/4})$ quantum queries are required to test expansion, ruling out the possibility of an exponential quantum speedup. This work naturally raises the question (also highlighted by Montanaro and de Wolf [14]) of whether there can ever be exponential quantum speedup for graph property testing, or for graph properties more generally.

[2] Aaronson and Ambainis required the function to be symmetric both under permuting the n bits of the input, and under permuting the alphabet symbols in Σ; Chailloux showed that the latter is not necessary.

B. Our contributions

In this work, we extend the results of Aaronson-Ambainis and Chailloux to other symmetry groups. We characterize general symmetries of functions as follows.

Definition 1. *Let $f\colon P \to \{0,1\}$ be a function with $P \subseteq \Sigma^n$, where Σ is a finite alphabet and $n \in \mathbb{N}$. We say that f is* symmetric *with respect to a permutation group G acting on domain $[n]$ if for all $x \in P$ and all $\pi \in G$, the string $x \circ \pi$ defined by $(x \circ \pi)_i := x_{\pi(i)}$ satisfies $x \circ \pi \in P$ and $f(x \circ \pi) = f(x)$.*

The case where $G = S_n$, the fully-symmetric permutation group, is the scenario handled in Chailloux's work [7]: he showed that $\mathrm{R}(f) = O(\mathrm{Q}(f)^3)$ if f is symmetric under S_n. Aaronson and Ambainis [6] required an even stronger symmetry property. We note that when Σ is large, say $|\Sigma| = n$ or larger, the class of functions symmetric under S_n is already highly nontrivial: among others, it includes functions such as COLLISION, an important function whose quantum query complexity was established in [15]; k-SUM, whose quantum query complexity required the negative-weight adversary to establish [16]; and k-DISTINCTNESS, whose quantum query complexity is still open [17].

In this work, we examine what happens when we relax the full symmetry S_n to smaller symmetry groups G. We introduce some tools for showing that particular classes of permutation groups G do not allow super-polynomial quantum speedups; that is, we provide tools for showing that every f symmetric with respect to G satisfies $\mathrm{Q}(f) = \mathrm{R}(f)^{\Omega(1)}$. Our first main result is the following theorem, in which G is the *graph symmetry*: the permutation group acting on strings of length $\binom{n}{2}$ (representing the possible edges of a graph), which includes all permutations of the edges induced by one of the $n!$ relabelings of the n vertices. Functions that take the adjacency matrix of a graph as input, and whose output depends only on the isomorphism class of the graph (not on the labeling of its vertices), are symmetric with respect to the graph symmetry G.

Theorem 2 (Informal version of Corollary 31)**.** *Any Boolean function f defined on the adjacency matrix of a graph (and symmetric with respect to renaming the vertices of the graph) has $\mathrm{R}(f) = O(\mathrm{Q}(f)^6)$. This holds even if f is a partial function.*

This theorem holds even when the alphabet of f is non-Boolean. We note that this is a strict generalization of the result of Aaronson and Ambainis [6], since any fully-symmetric function is necessarily symmetric under the graph symmetry as well. It is also a generalization of [7], except

that our polynomial degree (power 6) is larger than the power 3 of Chailloux. Our results also extend to other types of graph symmetries, including hypergraph symmetries and bipartite graph symmetries.

Next, we extend Theorem 2 to give a more general characterization of which classes of symmetries are inconsistent with super-polynomial quantum speedups. Our main result in this regard is a dichotomy theorem for *primitive* permutation groups. Primitive permutation groups are sometimes described as the "building blocks" of permutation groups, because arbitrary permutation groups can always be decomposed into primitive groups (in a certain formal sense). We give a complete classification of which primitive groups G allow super-polynomial quantum speedups and which do not, in terms of the *minimal base size* $b(G)$. The minimal base size is a key quantity in computational group theory that roughly captures the size of a permutation group G relative to the number of points it acts on. Thus, the following theorem amounts to showing that symmetries of "large" primitive permutation groups do not allow large quantum speedups, while symmetries of "small" primitive permutation groups do.

Theorem 3 (Informal version of Corollary 38)**.** *Let G be a primitive permutation group acting on $[n]$, and let $b(G)$ denote the size of a minimal base for G. If $b(G) = n^{\Omega(1)}$, then $R(f) = Q(f)^{O(1)}$ for every f that is symmetric under G. Otherwise, if $b(G) = n^{o(1)}$, then there exists a partial function f that is symmetric under G such that $R(f) = Q(f)^{\omega(1)}$.*

By decomposing an arbitrary permutation group into primitive groups, we can extend the above theorem to show that if a permutation group G does not allow super-polynomial speedups, then it must be constructed out of a constant number of primitive groups H that satisfy $b(H) = n^{\Omega(1)}$, where H acts on n points. In the other direction, if any of the primitive factors H of G satisfy $b(H) = n^{o(1)}$, or if G contains more than a constant number of primitive factors, then we exhibit a function that is symmetric under G with a super-polynomial quantum speedup.

Remarkably, the proof of Theorem 3 also includes a strong characterization of what the primitive permutation groups G that satisfy $b(G) = n^{\Omega(1)}$ look like. Indeed, we show that as a consequence of the classification of finite simple groups, all such groups essentially look like hypergraph symmetries or minor extensions thereof. Thus, in some sense, permutation groups built out of hypergraph symmetries are the *only* permutation groups that are inconsistent with super-polynomial speedups. We consider this one of the most surprising consequences of our work; a priori, one could expect there to be many different kinds of symmetries that disallow super-polynomial speedups.

Finally, we return to the topic of graph properties. While the aforementioned results show that no exponential speedup is possible for graph properties in the adjacency *matrix* model, they do not resolve the original open question of Ambainis, Childs, and Liu [13], which specifically addressed graph property testing in the adjacency *list* model. Graph symmetries in this model manifest themselves in a different way that is not captured by Definition 1, so Theorem 2 does not apply.[3] In this model, we show the following,

Theorem 4 (Informal version of Theorem 44)**.** *There exists a graph property testing problem in the adjacency list model for which there is an exponential quantum speedup.*

This is in stark contrast to the situation in the adjacency matrix case. Together, Theorem 2 and Theorem 4 fully settle the open question about the possibility of exponential quantum speedup for graph properties, showing that its answer is highly model-dependent: exponential speedup is impossible in the adjacency matrix model, but is possible in the adjacency list model, even in the restrictive setting of graph property *testing*.

C. Techniques

1) Well-shuffling symmetries do not allow speedups: Our analysis begins with a basic observation of Chailloux [7]. Suppose that f is symmetric under the full symmetric group S_n acting on $[n]$. Zhandry [18] showed that distinguishing a random element of S_n from a random small-range function $\alpha\colon [n] \to [n]$ with $|\alpha([n])| = r$ requires $\Omega(r^{1/3})$ quantum queries.[4] Now, if Q is a quantum algorithm solving f using T queries, then Q also outputs $f(x)$ on input $x \circ \pi$ (the input x with bits shuffled according to π) for any $\pi \in S_n$, since f is symmetric under S_n. In particular, $Q(x \circ \pi)$ for a random $\pi \in S_n$ still outputs $f(x)$. However, the T-query algorithm Q cannot distinguish a random $\pi \in S_n$ from a random function $\alpha\colon [n] \to [n]$ with range $r \approx T^3$. Therefore Q must output $f(x)$ with at most constant error even when run on $x \circ \alpha$ for a random small-range function α. This property can be used to simulate Q classically: a classical algorithm R can simply sample a small-range function α, explicitly query the entire string $x \circ \alpha$ (using only $O(T^3)$ queries since α has range $O(T^3)$), and then simulate Q on the string $x \circ \alpha$. This is an $O(T^3)$-query classical algorithm for computing f, created from a T-query quantum algorithm for f.

The above trick can be generalized from the fully-symmetric group S_n to other permutation groups G, provided we can show that it is hard for a T-query quantum

[3] Given a graph $x\colon [n] \times [d] \to [n] \cup \{*\}$ in the adjacency list model with n vertices and bounded degree d, its isomorphism class essentially consists of graphs of the form $\pi^{-1} \circ x \circ (\pi \times \mathrm{id}_{[d]})$ where $\pi \in S_n$ is a relabeling of vertices and π^{-1} is defined as the inverse of π but also maps $*$ to $*$. A function f is a graph property in this model if and only if its domain P is a union of such isomorphism classes and f is constant on each isomorphism class.

[4] Actually, we show that a version of Zhandry's result that is sufficient for our purposes follows easily from the collision lower bound, so his techniques are not necessary for our results.

algorithm to distinguish G from a function with range $\mathrm{poly}(T)$. The question of whether there exists an arbitrary symmetric function f with a quantum speedup is therefore reduced to the question of whether the concrete task of distinguishing G from the set of all small-range functions can be done quickly using a quantum algorithm. That is, if $D_{n,r}$ is the set of all strings in $[n]^n$ that use only r unique symbols, then we care about the quantum query cost of distinguishing $D_{n,r}$ from G; if this cost is $r^{\Omega(1)}$, then no function that is symmetric under G can exhibit a super-polynomial quantum speedup. In this case, we call G *well-shuffling*.

We show that the well-shuffling property is preserved under various operations one might perform on a permutation group, which allows us to prove that many permutation groups are well-shuffling simply by reduction to S_n. As an example, for undirected graph properties, the relevant transformation is from a permutation group G acting on a set $[n]$ to the induced action H on the $\binom{n}{2}$ unordered pairs of $[n]$. We then show that if G is well-shuffling, then H is also well-shuffling, because the problem of distinguishing G from a small range function has a reduction to the problem of distinguishing H from a small range function. More generally, this reduction works for ordered pairs or for tuples of any constant length.

2) Classification of symmetries with and without speedups: To characterize which permutation groups allow super-polynomial quantum speedups, we first show that large quantum speedups exist for any sufficiently "small" permutation group. In particular, if the minimal base size $b(G)$ of a permutation group G acting on n points satisfies $b(G) = n^{o(1)}$, then we exhibit a partial function that is symmetric under G with a super-polynomial quantum speedup. We construct such a function by starting with an arbitrary promise problem that has a sufficiently large separation between randomized and quantum query complexities (e.g., Simon's problem [2] or Forrelation [19]). We then modify it to be symmetric under G in a way that preserves the super-polynomial separation between randomized and quantum query complexities.

One might wonder whether a sort of converse holds, i.e., whether $b(G) = n^{\Omega(1)}$ implies that G is well-shuffling, and therefore that all partial functions symmetric under G admit at most a polynomial quantum speedup. We show this for primitive permutation groups G, essentially because the structure of "large" primitive groups is very well-understood. We base our proof on a theorem due to Liebeck [20] that characterizes minimal base sizes of primitive groups via the classification of finite simple groups. Liebeck's theorem allows us to show that primitive groups G with $b(G) = n^{\Omega(1)}$ can all be constructed out of the symmetric group via transformations that preserve the well-shuffling property. As a result, the quantity $b(G)$ completely characterizes whether a primitive permutation group admits a super-polynomial speedup. Moreover, the transformations involved are the same transformations we use to show that hypergraph symmetries are well-shuffling, so primitive groups that satisfy $b(G) = n^{\Omega(1)}$ all essentially look like hypergraph symmetries.

To go beyond primitive groups, we use the fact that an arbitrary permutation group can be decomposed into transitive groups (by looking at its orbits), which can then be decomposed into primitive groups (by looking at its nontrivial block systems). We show that for a group G, if any of the primitive factors in such a decomposition of G are consistent with super-polynomial speedups, or if the number of primitive factors is $\omega(1)$, then there exists a partial function that is symmetric under G with a super-polynomial quantum speedup. These functions are generally easy to construct, though in one case that involves tree-like symmetries, we make essential use of the 1-FAULT DIRECT TREES problem of Zhan, Kimmel, and Hassidim [21]: a partial function that has a super-polynomial quantum speedup, constructed by adding a promise to a Boolean evaluation tree in a way that preserves the symmetries of the tree. Interestingly, these results also show that the quantity $b(G)$ does *not* determine the well-shuffling property in general, because there exist permutation groups G acting on n points with $b(G) = \Omega(n)$ that are consistent with exponential quantum speedups. Nevertheless, we can at least say that if a permutation group G is inconsistent with super-polynomial quantum speedups, then it must be decomposable into $O(1)$ well-shuffling primitive groups.

3) Graph property testing in the adjacency list model: Finally, we show that the possibility of exponential quantum speedups for graph properties depends strongly on the input model: while the adjacency matrix model allows at most polynomial speedups, we give an example of an exponential speedup in the adjacency list model for bounded-degree graphs. In particular, we show that such a separation holds even in the more restrictive setting of property testing, where "no" instances are promised to be far from "yes" instances.

The main idea for this separation comes from work by Childs et al. [22], demonstrating an exponential algorithmic speedup by quantum walk. More specifically, they design a "welded trees" graph that consists of two depth-k binary trees welded together by an alternating cycle on the leaves of the two trees, with the two degree-2 vertices (roots) referred to as "ENTRANCE" and "EXIT". The main result of [22] is that a quantum computer, given the ENTRANCE, can find the EXIT in time $\mathrm{poly}(k)$, whereas a classical computer needs time $2^{\Omega(k)}$ to find the EXIT. However, it is not immediately clear how to lift this separation to the realm of graph property testing: it is not even a graph property (since the ENTRANCE is given and the EXIT can be labeled differently in isomorphic graphs), and it is still more challenging to find a related classically-hard property *testing* problem.

Our approach is to leverage the quantum advantage for

finding ENTRANCE-EXIT pairs to test whether the given graph has a certain structure derived from welded trees. The difficulty is that it is unclear how finding ENTRANCE-EXIT pairs helps a quantum computer test the structure of a welded trees. One of our main observations is that once the vertices in the weld of the binary trees are marked, it becomes easy to test the entire structure: we can separately test the two binary trees, and then test the weld itself. However, marking the weld vertices also enables a classical computer to find ENTRANCE-EXIT pairs, so it seems that this approach does not yield quantum advantage.

Our resolution is to use many copies of the welded trees structure and randomly assign a bit to every root (degree-2) vertex. Then we mark vertices by connecting every non-root vertex v to a root r via an "advice edge", such that

1) if v is in the weld, then the sum of the bits assigned to r and its pair is odd, and
2) if v is not in the weld, then the sum of the bits assigned to r and its pair is even.

To keep the degree bounded, instead of directly connecting advice edges to the roots, we attach a binary tree "antenna" to the roots and connect the advice edges there, as shown in Figs. 1 and 2.

Still, it might seem that this construction has a fatal chicken-or-egg problem: we need to mark the weld vertices to test the structure, but then marking the weld vertices enables also a classical computer to find the root pairs, leaving no quantum advantage. However, this is a feature, not a bug. The key point is that the marking of the weld vertices can only be read efficiently by a quantum computer, i.e., the information about the weld vertices is hidden such that it can only be retrieved efficiently by finding one root from another which can only be done efficiently by a quantum computer.

To show that this property is hard to test classically, we show that it is exponentially difficult to distinguish a randomly selected graph of the form described above, from a graph that looks like the above, except the binary trees are self-welded, i.e., the leaves of the binary trees are connected with a random cycle only containing vertices from the same binary tree (this then disconnects the root pairs, so the advice edges no longer have the original role and are chosen arbitrarily). The intuition behind our classical lower bound proof is the same as that behind the lower bound proof of [22]: given a root of a (self-)welded trees graph, the graph appears to be an exponentially deep binary tree for a (random) classical walker who can only explore the graph locally. Finally, we show that a random welded trees graph is $\Omega(1)$-far from a random self-welded trees graph by observing that the welded trees graph is bipartite, whereas the self-welded trees graph is $\Omega(1)$-far from being bipartite with very high probability.

D. Organization

We begin by introducing some preliminaries in Section II. In Section III we introduce the notion of well-shuffling permutation groups and discuss how they do not allow super-polynomial quantum speedups. Section IV introduces the property of well-shuffling, and shows that hypergraph symmetries cannot have super-polynomial quantum speedups. In Section V we classify well-shuffling permutation groups, showing that hypergraph symmetries are essentially the only groups that are inconsistent with super-polynomial speedups. Section VI presents an exponential separation between classical and quantum property testing in the adjacency list model for bounded-degree graphs. Finally, in Section VII we briefly discuss some open problems. Due to space constraints all the proofs are deferred to the full version of this work.[1]

II. Preliminaries

A. Query complexity

We begin by introducing some standard notation from query complexity. A *Boolean function* is a $\{0,1\}$-valued function f on strings of length $n \in \mathbb{N}$. We use $\mathrm{Dom}(f)$ to denote the domain of f, and we always have $\mathrm{Dom}(f) \subseteq \Sigma^n$ where Σ is a finite alphabet. The function f is called *total* if $\mathrm{Dom}(f) = \Sigma^n$; otherwise it is called *partial*.

For a (possibly partial) Boolean function f, we use $\mathrm{R}_\epsilon(f)$ to denote its *randomized query complexity with error* ϵ, as defined in [3]. This is the minimum number of queries required in the worst case by a randomized algorithm that computes f with worst-case error ϵ. We use $\mathrm{Q}_\epsilon(f)$ to denote the *quantum query complexity with error* ϵ of f, also defined in [3]. This is the minimum number of queries required in the worst case by a randomized algorithm that computes f with worst-case error ϵ. When $\epsilon = 1/3$, we omit it and simply write $\mathrm{R}(f)$ and $\mathrm{Q}(f)$.

B. Permutation groups

We review some basic definitions about permutation groups, following Hulpke [23].

Definition 5 (Permutation group)**.** *A* permutation group *is a pair (D, G) where D is a set and G is a set of bijections $\pi\colon D \to D$, such that G forms a group under composition (i.e., G contains the identity function and is closed under composition and inverse of the bijections). In this case, G is said to* act *on D. We will often denote a permutation group simply by G, with the domain D being implicit.*

In other words, a permutation group is simply a set of permutations of a domain D which is closed under composition and inverse. In this work we will generally take $D = [n] \coloneqq \{1, 2, \ldots, n\}$ for $n \in \mathbb{N}$. The set $[n]$ will represent the indices of an input string, or equivalently, the queries an algorithm is allowed to make.

We define orbits, transitivity, and stabilizers of permutation groups, all of which are standard.

Definition 6 (Orbit)**.** *Let G be a permutation group on domain D, and let $i \in D$. Then the* orbit *of i is the set $\{\pi(i) : \pi \in G\}$. A subset of D is an orbit of G if it is the orbit of some $i \in D$ with respect to G.*

Definition 7 (Transitivity)**.** *We say that a permutation group G on domain D is* k-transitive *if for all distinct $i_1, i_2, \ldots, i_k \in D$ and distinct $j_1, j_2, \ldots, j_k \in D$, there exists some $\pi \in G$ such that $\pi(i_t) = j_t$ for all $t = 1, 2, \ldots, k$. A group that is 1-transitive is called simply* transitive*.*

Definition 8 (Stabilizer)**.** *Let G be a permutation group on domain D. The* pointwise stabilizer *of a set $S \subseteq D$ is the subgroup $\mathrm{Stab}_G(S) \coloneqq \{\pi \in G : \pi(i) = i \ \forall i \in S\}$. The* setwise stabilizer *of a set $S \subseteq D$ is the subgroup $\mathrm{Stab}_G(\{S\}) \coloneqq \{\pi \in G : \pi(i) \in S \ \forall i \in S\}$.*

We use some facts about the structure of permutation groups. For this, we need some additional definitions, starting with the notions of primitive groups and wreath products.

Definition 9 (Block system)**.** *Let G be a permutation group on domain D. A partition $\mathcal{B} = \{B_1, B_2, \ldots, B_k\}$ of D is called a* block system *for G if $\mathcal{B}$ is G-invariant. Formally, this means that $\pi(B_i) \in \mathcal{B}$ for every $\pi \in G$ and $i \in [k]$, where $\pi(B_i) \coloneqq \{\pi(x) : x \in B_i\}$. The sets B_i are called* blocks *for G.*

It follows from this definition that if $\mathcal{B}$ is a block system for G, then for every block B_i and permutation $\pi \in G$, either $\pi(B_i) = B_i$ or $\pi(B_i) \cap B_i = \emptyset$. Moreover, if G acts transitively, then all blocks have the same size.

Definition 10 (Primitive group)**.** *Let G be a transitive permutation group on domain D. G is called* primitive *if the only block systems for G are $\{D\}$ and the partition of D into singletons. A permutation group that is not primitive is called* imprimitive*.*

Primitive groups are sometimes described as the "building blocks" of permutation groups, for reasons we will see later.

Definition 11. *Let G be an abstract group. The* direct power *of G with exponent m, denoted $G^{\times m}$, is the direct product of m copies of G : $G^{\times m} \coloneqq \underbrace{G \times \cdots \times G}_{m \text{ times}}$.*

If G is a permutation group acting on $[n]$, there are two natural ways to construct $G^{\times m}$ as a permutation group. Let $\pi = (\pi_1, \pi_2, \ldots, \pi_m) \in G^{\times m}$. The first action is on $[m] \times [n]$ where we have $\pi(i, j) = \pi_i(j)$. The second action is on $[n]^m$ where we have $\pi(i_1, i_2, \ldots, i_m) = (\pi_1(i_1), \pi_2(i_2), \ldots, \pi_m(i_m))$. These two actions of $G^{\times m}$ give rise to two actions of the *wreath product* of two groups:

Definition 12 (Wreath product)**.** *Let G and H be permutation groups acting on $[m]$ and $[n]$, respectively. The* wreath product *of G and H, denoted $G \wr H$, is one of two permutation groups: (i) The* imprimitive action *is the action on $[m] \times [n]$ generated by the natural action of $H^{\times m}$, along with permutations of the form $(i, j) \to (\sigma(i), j)$ for every $\sigma \in G$. (ii) The* product action *is the action on $[n]^m$ generated by the natural action of $H^{\times m}$, along with permutations of the form $(i_1, i_2, \ldots, i_m) \to (i_{\sigma(1)}, i_{\sigma(2)}, \ldots, i_{\sigma(m)})$ for every $\sigma \in G$.*

Fact 13. *Let G and H be permutation groups acting on $[m]$ and $[n]$, respectively. If $n > 1$, then the imprimitive action and the primitive action of $G \wr H$ are isomorphic as groups, and both have order $|G| \cdot |H|^m$.*

If it is clear from context which of the two actions we are referring to, then we may sometimes simply write $G \wr H$ without specifying the type of action. The wreath product imprimitive action carries a useful intuitive interpretation: if g and h are functions that are symmetric under permutation groups G and H, respectively, then $G \wr H$ is the (visible)[5] group of symmetries of the block-composed function $g \circ h$. Thus, the wreath product imprimitive action is associative. Note that the wreath product is not commutative.[6]

Definition 14 (Base)**.** *Let G be a permutation group on domain D. A* base *for G is a set $S \subseteq D$ such that the pointwise stabilizer $\mathrm{Stab}_G(S)$ is the trivial group. The* minimal base size *for G, denoted $b(G)$, is the size of the smallest base for G.*

Bases of permutation groups are important in computational group theory for the following reason: if S is a base for G, then a permutation $\pi \in G$, viewed as a function $D \to D$, is uniquely determined by its restriction to S. This can be useful particularly when S is small compared to D.

In many applications, the minimal base size is a useful proxy for the size of a permutation group. This is quantified by the following well-known lemma:

Lemma 15. *Let G be a permutation group on $[n]$. Then the order of the group $|G|$ and the minimal base size $b(G)$ satisfy $2^{b(G)} \leq |G| \leq n^{b(G)}$. Equivalently, $\log_n(|G|) \leq b(G) \leq \log_2(|G|)$.*

C. Symmetric functions

We introduce some notation that is used throughout the paper to talk about symmetric functions.

Definition 16 (Permuting strings)**.** *Let π be a permutation on $[n]$, and let $x \in \{0,1\}^n$. We write $x \circ \pi$ to denote the string whose characters have been permuted by π; that is, $(x \circ \pi)_i \coloneqq x_{\pi(i)}$. More generally, $x \circ \pi$ is similarly defined when π is merely a function $[n] \to [n]$ rather than a permutation.*

[5] In rare cases, depending on g and h, there can be more symmetries. For example, let $g = h = \mathsf{OR}_n$, which both have symmetry group S_n. The wreath product $S_n \wr S_n$ is *not* the same as S_{n^2}, which is the group of symmetries of $g \circ h = \mathsf{OR}_{n^2}$. However, we can say that $g \circ h$ is guaranteed to be *at least* symmetric under $G \wr H$ for any functions g and h.

[6] This can be confusing because some authors instead denote this wreath product as $H \wr G$. We prefer the notation used here because of the correspondence to block composition of functions.

Note that if we view a string $x \in \Sigma^n$ as a function $[n] \to \Sigma$ with $x(i) \coloneqq x_i$, then $x \circ \pi$ is simply the usual function composition of x and π. This notation allows us to easily define symmetric functions.

Definition 17 (Symmetric function)**.** *Let G be a permutation group on $[n]$, and let f be a (possibly partial) Boolean function with $\mathrm{Dom}(f) \subseteq \Sigma^n$. We say f is* symmetric under *G if for all $x \in \mathrm{Dom}(f)$ and all $\pi \in G$ we have $x \circ \pi \in \mathrm{Dom}(f)$ and $f(x \circ \pi) = f(x)$.*

For asymptotic bounds such as $\mathrm{Q}(f) = \mathrm{R}(f)^{\Omega(1)}$ to be well-defined, we need to talk about *classes* of functions rather than individual functions. To do that, we need to talk about classes of permutation groups. We introduce the following definition, which defines, for a class of permutation groups $\mathcal{G}$, the set of all functions symmetric under some group in $\mathcal{G}$. We denote this set by $F(\mathcal{G})$.

Definition 18 (Class of symmetric functions)**.** *Let $\mathcal{G} = \{G_i\}_{i \in I}$ be a (possibly infinite) set of finite permutation groups, with G_i acting on $[n_i]$ for each $i \in I$. Here I is an arbitrary index set and $n_i \in \mathbb{N}$ for all $i \in I$. Then define $F(\mathcal{G})$ to be the set of all (possibly partial) Boolean functions that are symmetric under some G_i. That is, we have $f \in F(\mathcal{G})$ if and only if $f\colon \mathrm{Dom}(f) \to \{0,1\}$ is a function with $\mathrm{Dom}(f) \subseteq [m]^n$ for some $n, m \in \mathbb{N}$, and f is symmetric under G_i for some $i \in I$ such that $n_i = n$. (Here $[m]$ represents the alphabet Σ.)*

III. Well-shuffling permutation groups

We first define the notion of a well-shuffling class of permutation groups, which is a class $\mathcal{G}$ of permutation groups G that are hard to distinguish from the set of small-range functions via a quantum query algorithm. We then show that a well-shuffling class of permutation groups does not allow super-polynomial quantum speedups.

We start by defining the set of small-range strings $D_{n,r}$.

Definition 19 (Small-range strings)**.** *For $n, r \in \mathbb{N}$, let $D_{n,r}$ be the set of all strings α in $[n]^n$ for which the number of unique alphabet symbols in α is at most r.*

We identify a string $\alpha \in [n]^n$ with a function $[n] \to [n]$. Then $D_{n,r}$ is the set of all functions $[n] \to [n]$ with range size at most r. Next, we define $\mathrm{cost}(G, r)$ as the quantum query complexity of distinguishing G from $D_{n,r}$ (where G is a permutation group acting on $[n]$).

Definition 20 (Cost)**.** *Identify a permutation on $[n]$ with a string in $[n]^n$ in which each alphabet symbol occurs exactly once. Then a permutation group G on $[n]$ corresponds to a subset of $[n]^n$. For $r < n$, let $\mathrm{cost}_\epsilon(G, r)$ be the minimum number of quantum queries needed to distinguish G from $D_{n,r}$ to worst-case error ϵ; that is, $\mathrm{cost}_\epsilon(G, r) \coloneqq \mathrm{Q}_\epsilon(f)$, where f has domain $G \cup D_{n,r} \subseteq [n]^n$ and is defined by $f(x) = 1$ if $x \in G$ and $f(x) = 0$ if $x \in D_{n,r}$. When $r \geq n$, we set $\mathrm{cost}_\epsilon(G, r) \coloneqq \infty$. When $\epsilon = 1/3$, we omit it and write $\mathrm{cost}(G, r)$.*

We note that since $\mathrm{cost}_\epsilon(G, r)$ is defined as the worst-case quantum query complexity of a Boolean function, it satisfies amplification, meaning that the precise value of ϵ does not matter so long as it is a constant in $(0, 1/2)$ and so long as we do not care about constant factors.

We define a well-shuffling class of permutation groups as follows.

Definition 21 (Well-shuffling permutation groups)**.** *Let $\mathcal{G}$ be a collection of permutation groups. We say $\mathcal{G}$ is* well-shuffling *if $\mathrm{cost}(G, r) = r^{\Omega(1)}$ for $G \in \mathcal{G}$ and $r \in \mathbb{N}$. More explicitly, we say $\mathcal{G}$ is well-shuffling* with power $a > 0$ *if there exists $b > 0$ such that $\mathrm{cost}(G, r) \geq r^{1/a}/b$ for all $G \in \mathcal{G}$ and all $r \in \mathbb{N}$.*

We note that $\mathrm{cost}(G, r) \geq r^{1/a}/b$ is always satisfied when r is greater than or equal to the domain size of G, since in that case $\mathrm{cost}(G, r) = \infty$. Hence to show well-shuffling we only need to worry about r smaller than n, the domain size of the permutation group G.

The following theorem plays a central role in this work: it shows that a well-shuffling collection of permutation groups does not allow super-polynomial quantum speedups.

Theorem 22. *Let $f\colon \mathrm{Dom}(f) \to \{0,1\}$ be a partial Boolean function on $n \in \mathbb{N}$ bits, with $\mathrm{Dom}(f) \subseteq \Sigma^n$ (where Σ is a finite alphabet). Let G be a permutation group on $[n]$, and suppose that f is symmetric under G. Then there is a universal constant $c \in \mathbb{N}$ such that*

$$\mathrm{R}(f) \leq \min\{r \in \mathbb{N} : \mathrm{cost}(G, r) \geq c\,\mathrm{Q}(f)\}.$$

Consequently, if $\mathcal{G}$ is a well-shuffling collection of permutation groups with power a, then for all $f \in F(\mathcal{G})$ we have $\mathrm{R}(f) = O(\mathrm{Q}(f)^a)$.

The upshot of Theorem 22 is that we can show a class of permutation groups $\mathcal{G}$ does not allow super-polynomial quantum speedups simply by showing that it is well-shuffling—that is, by showing that $G \in \mathcal{G}$ is hard to distinguish from the set of small-range functions $D_{n,r}$ using a quantum query algorithm.

IV. Showing permutation groups are well-shuffling

In this section, we introduce some tools for showing that a collection of permutation groups is well-shuffling. Using Theorem 22 these can be directly used to show that certain symmetries are not consistent with large quantum speedups.

A. The symmetric group

The first fundamental result is that the class of full symmetric permutation groups S_n is well-shuffling. This was shown by Zhandry [18] in a different context, though we also

provide a simpler proof by a reduction from the collision problem.[7]

Theorem 23. *There is a universal constant C such that any quantum algorithm distinguishing a permutation in S_n from a string in $D_{n,r}$ must make at least $r^{1/3}/C$ queries.*

This theorem says that S_n is hard to distinguish from $D_{n,r}$. Since we prove Theorem 23 by a reduction from collision, and since our main tools from here on are additional reductions, effectively all lower bounds in this paper work by reductions from collision. From Theorem 23, the following two corollaries immediately follow (in light of Theorem 22).

Corollary 24. *The set of symmetric groups $\mathcal{S} = \{S_n\}_{n\in\mathbb{N}}$ is well-shuffling with power 3.*

Corollary 25. *All (possibly partial) Boolean functions f that are symmetric under the full symmetric group S_n satisfy $\mathrm{R}(f) = O(\mathrm{Q}(f)^3)$.*

B. Transformations for graph symmetries

Next, we introduce some additional transformations on permutation groups that approximately preserve the cost. These allows us to show that graph property permutation groups (and several variants of them) are well-shuffling.

1) Transformation for directed graphs: We start by defining an extension of a permutation group G on $[n]$ to a permutation group acting on $[n]^\ell$. The notation in the definition below comes from [24].

Definition 26. *Let G be a permutation group on domain D, and let $\ell \in \mathbb{N}$. Define $G^{(\ell)}$ to be the permutation group that acts on domain D^ℓ by $\pi(i_1, i_2, \ldots, i_\ell) = (\pi(i_1), \pi(i_2), \ldots, \pi(i_\ell))$ for each $\pi \in G$ (so the number of permutations in $G^{(\ell)}$ is the same as the number of permutations in G).*

Define $G^{\langle\ell\rangle}$ to be the permutation group $G^{(\ell)}$ with domain restricted to the subset $D^{\langle\ell\rangle} \subseteq D^\ell$ consisting of all distinct ℓ-tuples of elements of D.

We show that both of these transformations preserve the cost, at least when ℓ is constant. When G is a permutation group on $[n]$, and H is the permutation group $G^{(\ell)}$, the $\mathrm{cost}(H, r^\ell) \geq \mathrm{cost}(G, r)/\ell$. To handle $G^{\langle\ell\rangle}$, we show that restricting the domain of a permutation group to some union of its orbits does not decrease its cost, i.e., $\mathrm{cost}(G', r) \geq \mathrm{cost}(G, r)$. Here G' is the permutation group G acting only on a union of orbits of G. From these it follows that if G is a permutation group on $[n]$, and H is the permutation group $G^{\langle\ell\rangle}$ then $\mathrm{cost}(H, r^\ell) \geq \mathrm{cost}(G, r)/\ell$. Then the transformation $G^{\langle\ell\rangle}$ immediately allows us to show that directed graph symmetries are well-shuffling.

[7]Zhandry [18] showed that the hard distribution over $D_{n,r}$ is uniform, but we do not need this fact.

Corollary 27 (Directed graph symmetries)**.** *The set $\mathcal{G} = \{G_k\}_{k\in\mathbb{N}}$ of all directed graph symmetries is well-shuffling with power 6. Here the permutation group G_k acts on a domain of size $n = k(k-1)$ representing the possible arcs of a k-vertex directed graph, and G_k consists of all $k!$ permutations on these arcs that act by relabeling the vertices.*

The collection of directed hypergraph symmetries is similarly well-shuffling.

Corollary 28 (Directed hypergraph symmetries)**.** *The set $\mathcal{G}_p = \{G_k\}_{k\in\mathbb{N}}$ consisting of all p-uniform directed hypergraph symmetries is well-shuffling with power $3p$.*

2) Transformation for undirected graphs: To handle undirected graphs, we introduce yet another operation on permutation groups which approximately preserves the cost.

Theorem 29. *Let G be a permutation group, and let $\mathcal{B} = \{B_1, B_2, \ldots, B_k\}$ be a block system for G, where the blocks have equal size. Let H be the permutation group on $[k]$ induced by the action of G on the blocks. Then $\mathrm{cost}(H, r) \geq \mathrm{cost}(G, r)$.*

Using the above theorem we show that the collection of (undirected) graph symmetries is well-shuffling.

Definition 30 (Graph Symmetries)**.** *The collection of* graph symmetries *is the set $\mathcal{G} = \{G_k\}_{k\in\mathbb{N}}$ of permutation groups with G_k acting on $[n]$ with $n = k(k-1)/2$, such that the domain $[n]$ represents the set of all possible edges in a k-vertex graph, and G_k acts on these edges and permutes them in a way that corresponds to relabeling the vertices of the underlying graph.*

Corollary 31. *The set of all graph symmetries is well-shuffling with power 6. Hence $\mathrm{R}(f) = O(\mathrm{Q}(f)^6)$ for functions f symmetric under a graph symmetry.*

Using similar arguments of Corollary 28, we can show a similar result for undirected hypergraphs.

Corollary 32. *For every constant $p \in \mathbb{N}$, the collection of all p-uniform hypergraph symmetries is well-shuffling with power $3p$.*

Moreover, by introducing yet more operations on permutation groups for the case of bipartite graph symmetries, we can show that the bipartite graph symmetries are also well-shuffling.

V. Classification of well-shuffling permutation groups

In this section, we show that the results from the previous section are qualitatively optimal, in the sense that permutation groups constructed out of hypergraph symmetries are essentially the *only* groups that are not consistent with super-polynomial quantum speedups. Our starting point is an

observation that super-polynomial quantum speedups can be constructed out of any permutation group with sufficiently small minimal base size.

Proposition 33. *Let G be a permutation group on $[n]$, and let f be a (possibly partial) Boolean function with $\mathrm{Dom}(f) \subseteq \Sigma^n$ for some alphabet Σ. Then there exists a partial Boolean function g that is symmetric under G such that $Q(g) \leq Q(f) + b(G)$ and $R(g) \geq R(f)$.*

Put another way, if $b(G) = n^{o(1)}$, then by choosing an arbitrary function f with $Q(f) = n^{o(1)}$ and $R(f) = n^{\Omega(1)}$ (e.g., Simon's problem [2] or Forrelation [19]), then one can construct a function g symmetric under G with $Q(g) = n^{o(1)}$ and $R(g) = n^{\Omega(1)}$. Thus, permutation groups that do not allow super-polynomial speedups must have large minimal base size. Naturally, the question is does the converse holds. We show that it holds for primitive permutation groups and moreover such groups with $b(G) = n^{\Omega(1)}$ all look roughly like symmetries of p-uniform hypergraphs (Corollary 37). This completely classifies all primitive permutation groups with respect to super-polynomial quantum speedups.

A. Primitive groups

We use the following theorem due to Liebeck [20] which shows that there is a tight connection between minimal base size and the structure of the group. Its proof relies on the classification of finite simple groups.

Theorem 34. *Let G be a primitive permutation group on $[n]$. Then one of the following holds:*

(i) G is a subgroup of $S_\ell \wr S_m$ containing $A_m^{\times \ell}$, where the action of S_m is on p-element subsets of $[m]$ and the wreath product has the product action of degree $n = \binom{m}{p}^\ell$, or

(ii) $b(G) < 9 \log_2 n$.

Here, S_n denotes the symmetric group on $[n]$ and A_n denotes the alternating group on $[n]$.

Observe that in case (i), when $\ell = 1$, this corresponds precisely to either the set of p-uniform undirected hypergraph symmetries, or the index-2 subgroup of even permutations of the vertices. As long as both ℓ and p are constant, then these groups are well-shuffling:

Corollary 35. *For all constant ℓ, p, the collection of primitive permutation groups that satisfy case (i) of Theorem 34 is well-shuffling with power $3\ell p$.*

In fact, the following theorem shows that a collection of primitive permutation groups is well-shuffling if and only if the permutation groups are as in case (i) of Theorem 34, with ℓ and p both constant:

Theorem 36. *For all constant $\epsilon > 0$, there exists $c \in \mathbb{N}$ such that the collection of primitive permutation groups G with $b(G) \geq n^\epsilon$ is well-shuffling with power c. Moreover, for all sufficiently large n, all such G are as in case (i) of Theorem 34 with $\ell p \leq c/3$.*

Corollary 37. *Let $\mathcal{G}$ be a collection of primitive permutation groups. The following are equivalent:*

(i) $\mathcal{G}$ is well-shuffling.

(ii) $b(G) = n^{\Omega(1)}$ for $G \in \mathcal{G}$ acting on $[n]$.

(iii) All but finitely many $G \in \mathcal{G}$ satisfy case (i) of Theorem 34 with $\ell p = O(1)$.

Corollary 38. *Let $\mathcal{G}$ be a collection of primitive permutation groups. Then:*

(i) If $b(G) = n^{\Omega(1)}$ for $G \in \mathcal{G}$ acting on $[n]$, then $\mathcal{G}$ is well-shuffling, and thus $R(f) = Q(f)^{O(1)}$ for all $f \in F(\mathcal{G})$.

(ii) If $b(G) = n^{o(1)}$ for $G \in \mathcal{G}$ acting on $[n]$, then there exists a class of partial functions $f \in F(\mathcal{G})$ with $R(f) = Q(f)^{\omega(1)}$.

B. Imprimitive groups

Here we will see that there exist permutation groups G with $b(G) = n^{\Omega(1)}$ that are consistent with large quantum speedups. So, $b(G) = n^{\Omega(1)}$ is not equivalent to the well-shuffling property in general. Nevertheless, we can show that primitive permutation groups with $b(G) = n^{\Omega(1)}$ are the "building blocks" of well-shuffling group actions. Indeed, we show in Corollary 43 that if a collection of imprimitive permutation groups is inconsistent with super-polynomial speedups, then it must be built out of a constant number of well-shuffling primitive groups. Thus, in some sense, all permutation groups that do not allow large quantum speedups must involve symmetries of p-uniform hypergraphs. To prove these, we start by showing that quantum speedups can be constructed out of any permutation group with many orbits.

Proposition 39. *Let G be a permutation group on $[n]$. Let $\mathcal{O} = \{O_1, O_2, \ldots, O_k\}$ be a partition of $[n]$ into k orbits of G. Let f be a (possibly partial) Boolean function with $\mathrm{Dom}(f) \subseteq \Sigma^k$ for some alphabet Σ. Then there exists a partial Boolean function g that is symmetric under G such that $Q(g) = Q(f)$ and $R(g) = R(f)$.*

Thus by choosing f to be some query problem with $Q(f) = O(1)$ but $R(f) = \omega(1)$ (e.g., Forrelation [19]), one can construct a super-polynomial speedup out of any collection of permutation groups with $\omega(1)$ orbits.

Next, we observe that if G restricted to any orbit is consistent with super-polynomial speedups, then so is G as a whole.

Proposition 40. *Let G be a permutation group on $[n]$. Let $\mathcal{O} = \{O_1, O_2, \ldots, O_k\}$ be a partition of $[n]$ into k orbits of G, and let $G|_{O_i}$ denote the restriction of G acting only on O_i. Let f be a (possibly partial) Boolean function that is symmetric under $G|_{O_i}$ with $\mathrm{Dom}(f) \subseteq \Sigma^{|O_i|}$ for some alphabet Σ and orbit i. Then there exists a partial Boolean*

function g that is symmetric under G such that $Q(g) = Q(f)$ and $R(g) = R(f)$.

Thus, a collection of permutation groups that does not allow any super-polynomial quantum speedups must remain so when restricted to any subset of its orbits. For this reason, the main task is to understand which *transitive* permutation groups (i.e., groups with a single orbit) allow super-polynomial quantum speedups. We use the well-known embedding theorem by [23] (See Theorem II.49 of [23]) which shows that impritive transitive groups can be expressed as subgroups of a wreath product constructed from a nontrivial block system. Applying this recursively, one can always decompose a transitive imprimitive permutation group as a subgroup of an iterated wreath product of primitive groups. Thus, primitive groups are sometimes described as the "building blocks" of permutation groups.

The next proposition shows that if any terms in such a wreath product decomposition are consistent with super-polynomial speedups, then so is the group as a whole.

Proposition 41. *Let G be a permutation group on $[n]$ that is a subgroup of an iterated wreath product $G_1 \wr G_2 \wr \cdots \wr G_d$ in imprimitive action. Suppose f is a (possibly partial) function symmetric under G_i for some i. Then there exists a partial function g symmetric under G with $Q(g) = Q(f)$ and $R(g) = R(f)$.*

Less trivially, if a wreath product decomposition of a group is sufficiently deep, then the group is also consistent with large speedups. The following construction is based on the "1-FAULT DIRECT TREES" problem of [21], [25]:

Theorem 42. *Let G be a permutation group on $[n]$ that is a subgroup of an iterated wreath product of nontrivial permutation groups $G_1 \wr G_2 \wr \cdots \wr G_d$ in imprimitive action. Then there exists a function f symmetric under G such that $Q(f) = O(1)$ but $R(f) = \Omega(\log d)$.*

We remark that Theorem 42 shows the existence of super-polynomial speedups under permutation groups with large base. Indeed, if we define $G = \underbrace{S_2 \wr S_2 \wr \cdots \wr S_2}_{d \text{ times}}$ to be the depth-d iterated wreath product in imprimitive action of the symmetric group S_2, then $b(G) = 2^{d-1}$ even though G acts on 2^d points.

Putting all of these together, we can say

Corollary 43. *Let $\mathcal{G}$ be a collection of permutation groups. Suppose that for every $f \in F(\mathcal{G})$, we have $R(f) = Q(f)^{O(1)}$. Then the following must hold:*

(i) Permutation groups in $\mathcal{G}$ have $O(1)$ orbits.

(ii) The collection of restrictions of groups $G \in \mathcal{G}$ to subsets of their orbits is also inconsistent with super-polynomial quantum speedups.

(iii) All wreath product decompositions of groups $G \in \mathcal{G}$ restricted to an orbit have $O(1)$ primitive factors.

(iv) The collection of primitive factors that appear in such wreath product decompositions is well-shuffling. Equivalently, by Corollary 37, those primitive factors G that act on $[n]$ have $b(G) = n^{\Omega(1)}$, and all but finitely many satisfy case (i) of Theorem 34 with $\ell p = O(1)$.

VI. EXPONENTIAL SPEEDUP FOR ADJACENCY-LIST GRAPH PROPERTY TESTING

Having shown that graph properties in the adjacency *matrix* model cannot have exponential quantum speedup, we now turn to the adjacency *list* model. Specifically, we describe a graph property testing problem in the adjacency list model for which there is an exponential quantum speedup. We work with graphs that are undirected and have degree at most 5, while for convenience[8] we allow the graphs to have self-loops and parallel edges (we count these with multiplicity toward the degree bound).

Recall that in property testing, the goal is to distinguish between some set of yes instances and the set of no instances consisting of all graphs that are ε-far from the yes instances. More precisely we say that a graph $G = (V, E)$ (with maximum degree at most 5) is ε-far from the property $\mathcal{P}$, if for any yes instance $G' = (V, E') \in \mathcal{P}$ with n vertices, the symmetric difference of the edge sets has size $|E \triangle E'| \geq \varepsilon n$, where we count edges with multiplicity. Since we work with graph properties, we also require that $\mathcal{P}$ is invariant under permuting the vertices.

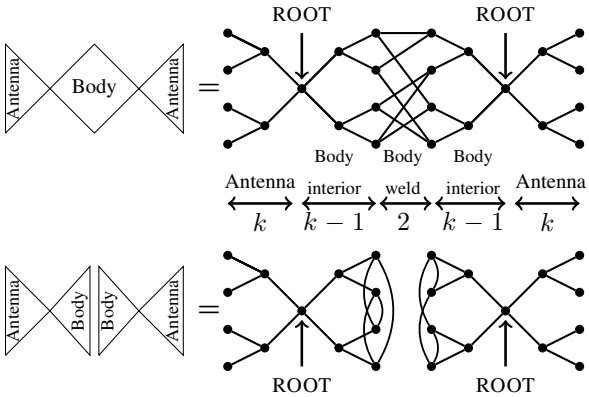

Figure 1: An illustration of a candy graph in the yes instance (top right) and our shorthand symbol for it (top left), and a double-bow-tie graph which appears in the particular no instances considered for our classical lower bound proof (bottom right) and our shorthand symbol for it (bottom left).

The graph property $\mathcal{P}_k$ that we want to test is illustrated in Figure 2 and can be described as follows. A graph has the property $\mathcal{P}_k$ if its single-edges form a graph that contains

[8]This is simply for clarity of presentation; we use self-loops and double edges as markers, for nodes and edges respectively. However, one could also work with simple graphs only and get the same exponential separation. This can be achieved by using slightly more complicated marker structures, e.g., one could replace self-loops with an edge connected to a triangle, etc.

$2^k - 1$ disconnected instances of "candy" graphs as shown in Figure 1 (top). A candy graph is obtained from 4 binary trees of depth k: two "antenna" trees A_1, A_2 and two "body" trees B_1, B_2. The candy graph is formed by first connecting the leaves of B_1 and B_2 by a collection[9] of alternating cycles containing all leaves of the body trees, and then merging the roots of $A_1 - B_1$ and $A_2 - B_2$. Additionally, in the full graph, roughly half of the candy (sub)graphs get a self-loop on exactly one of the roots, and the other candy graphs have either no self-loops or both roots get a self-loop.[10] There is also a double edge attached to every non-root vertex of the candy graphs, namely every body vertex is connected to a distinct antenna vertex by a double edge in the following way: if the body vertex is a "weld" vertex (i.e., it was a leaf of a body tree), then it gets attached to an antenna vertex inside a candy graph that has exactly one root with self-loop, while every interior vertex (i.e., non-weld and non-root) gets attached to an antenna vertex in a candy graph where the parity of the number of self-loops is even.

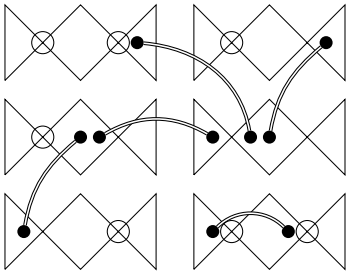

Figure 2: Multiple candy (sub)graphs which together form a yes-instance graph with property $\mathcal{P}_k$. In the particular no instances we consider, the candy graphs are replaced by double-bow-tie graphs. Here, we only show six of them and only five of the many "advice edges" (indicated by double 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 in the interior or the weld. An even parity of circles indicates interior, while an odd parity indicates weld.

While we do not claim that testing $\mathcal{P}_k$ is particularly useful, we can prove that a quantum computer has an exponential advantage in testing it.

Theorem 44. *In the adjacency list model, there exists a constant ϵ such that testing whether a bounded-degree graph has property $\mathcal{P}_k$ or is ϵ-far from having it can be done by a quantum algorithm using $Q(\mathcal{P}_k) = \mathrm{poly}(k)$ queries, with constant success probability and perfect completeness. On the other hand, any classical randomized algorithm that can test the property with bounded two-sided error needs $R(\mathcal{P}_k) = \exp(\Omega(k))$ queries.*

To prove this theorem we first prove that, given advice on which vertices are weld vertices, $\mathcal{P}_k$ can be tested *classically* with perfect completeness and constant success probability using $\mathrm{poly}(k)$ queries. The main idea is that there is a classical algorithm using $\mathrm{poly}(k)$ queries that can test whether a graph is a full binary tree of depth k, whose leaves are "marked" by being the only vertices of degree 1. We can use this algorithm to test $\mathcal{P}_k$ first disregarding the advice edges. Then the advice edges can be tested separately, since the advice also helps revealing ROOT-ROOT pairs in the candy graphs – ultimately verifying the given advice itself.

Next we show that for graphs with $\mathcal{P}_k$ the advice at each vertex v can be computed on a quantum computer with probability 1 using $\mathrm{poly}(k)$ queries. This is essentially by construction: the quantum walk algorithm of Childs et al. [22] efficiently finds the ROOT-ROOT pairs in the candy graph connected to v by advice edges, revealing the advice.

For classical hardness, we show that randomly chosen graphs from two families of some "hard" yes and no instances are exponentially difficult to distinguish on a classical computer. The particular yes instances consist of graphs with $\mathcal{P}_k$ that have single cycles welding together the candy graphs. The particular no instances are just like the yes instances, except that each candy graph is modified to look like a double-bow-tie graph as illustrated in Figure 1 (bottom). Noting that candy graphs forming graphs in $\mathcal{P}_k$ are bipartite, we can show that the considered no instances are typically far from having $\mathcal{P}_k$ by showing that (all but a vanishing proportion of) double-bow-tie graphs are far from being bipartite. On the other hand, we can also show that, to a classical computer, both candy and double-bow-tie graphs look like the same deep binary tree unless $\exp(\Omega(k))$ queries have been made. This establishes the classical hardness.

VII. Open problems

Finally we conclude by mentioning a few open problems.

- We prove that p-uniform hypergraph properties have at most a power $3p$ separation between quantum and randomized query complexity. However, we do not know if this is tight.
- What is the largest possible separation between $Q(f)$ and $R(f)$ for p-uniform hypergraph properties f? That is, what is the largest k for which there exists such an f with $R(f) = \Omega(Q(f)^k)$?
- In Corollary 43, we showed that well-shuffling permutation groups must be constructed out of a constant number of primitive groups with sufficiently large minimal base size. We conjecture that a converse holds, which would imply a complete dichotomy regard-

[9] In the lower bound proof we assume there is a single long cycle, but this is hard to test, so for the definition of the property we allow smaller cycles as well.

[10] Strictly speaking, a $2^k/(2^{k+1} - 2)$ fraction of candy graphs should get exactly one self-loop and the remaining fraction should be split evenly between getting two or zero self-loops, because there are $2 \cdot 2^k$ weld vertices while only $2 \cdot (2^k - 2)$ interior vertices in each candy graph, cf. the following main text. With a slight modification to the construction this could be balanced out, for example by replacing the roots by a path of length 5, and treating the middle vertex of this length-5 path as the new root.

ing which permutation groups allow super-polynomial quantum speedups and which do not:

Conjecture 45. *Let $\mathcal{G}$ be a collection of permutation groups that satisfies conditions (i)–(iv) of Corollary 43. Then $\mathcal{G}$ is well-shuffling, and thus $R(f) = Q(f)^{O(1)}$ for every $f \in F(\mathcal{G})$.*

- In Proposition 33 our construction of super-polynomial quantum speedups out of any sufficiently small permutation group requires large alphabets. Is it possible to construct such functions over a smaller alphabet?
- Finally, it remains open to find an adjacency-list graph property testing problem of *practical interest* that exhibits an exponential quantum speedup.

Acknowledgements

We thank Scott Aaronson and Carl Miller for many helpful discussions. SP also thanks Zak Webb for many related discussions. Part of this work was done while visiting the Simons Institute for the Theory of Computing; we gratefully acknowledge its hospitality. AMC and DW acknowledge support from the Army Research Office (grant W911NF-20-1-0015); the Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Quantum Algorithms Teams and Accelerated Research in Quantum Computing programs; and the National Science Foundation (grant CCF-1813814). AG acknowledges funding provided by Samsung Electronics Co., Ltd., for the project "The Computational Power of Sampling on Quantum Computers". Additional support was provided by the Institute for Quantum Information and Matter, an NSF Physics Frontiers Center (NSF Grant PHY-1733907). WK acknowledges support from a Vannevar Bush Fellowship from the US Department of Defense.

References

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science (FOCS)*, 1994, pp. 124–134, arXiv:quant-ph/9508027

[2] D. R. Simon, "On the power of quantum computation," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1474–1483, 1997.

[3] H. Buhrman and R. de Wolf, "Complexity measures and decision tree complexity: a survey," *Theory of Computing*, vol. 288, no. 1, pp. 21–43, 2002.

[4] R. Cleve, "The query complexity of order-finding," *Information and Computation*, vol. 192, no. 2, pp. 162–171, 2004, arXiv:quant-ph/9911124

[5] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf, "Quantum lower bounds by polynomials," *Journal of the ACM*, vol. 48, no. 4, pp. 778–797, 2001, earlier version in FOCS'98. arXiv:quant-ph/9802049

[6] S. Aaronson and A. Ambainis, "The need for structure in quantum speedups," *Theory of Computing*, vol. 10, pp. 133–166, 2014, arXiv:0911.0996

[7] A. Chailloux, "A note on the quantum query complexity of permutation symmetric functions," in *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*, 2018, pp. 19:1–19:7, arXiv:1810.01790

[8] S. Ben-David, "The structure of promises in quantum speedups," in *Proceedings of the 11th Conference on the Theory of Quantum Computation, Communication, and Cryptography (TQC)*, 2016, pp. 7:1–7:14, arXiv:1409.3323

[9] S. Aaronson and S. Ben-David, "Sculpting quantum speedups," in *Proceedings of the 31st IEEE Conference on Computational Complexity (CCC)*, 2016, pp. 26:1–26:28, arXiv:1512.04016

[10] O. Goldreich, S. Goldwasser, and D. Ron, "Property testing and its connection to learning and approximation," *Journal of the ACM*, vol. 45, no. 4, pp. 653–750, 1998.

[11] N. Alon and M. Krivelevich, "Testing k-colorability," *SIAM Journal on Discrete Mathematics*, vol. 15, no. 2, pp. 211–227, 2002.

[12] O. Goldreich and D. Ron, "Property testing in bounded degree graphs," in *Proceedings of the 29th ACM Symposium on the Theory of Computing (STOC)*, 1997, p. 406–415.

[13] A. Ambainis, A. M. Childs, and Y.-K. Liu, "Quantum property testing for bounded-degree graphs," in *Proceedings of the 15th International Workshop on Randomization and Computation (RANDOM)*, ser. Lecture Notes in Computer Science. Springer, 2011, vol. 6845, pp. 365–376, arXiv:1012.3174

[14] A. Montanaro and R. de Wolf, *A survey of quantum property testing*, ser. Graduate Surveys. Theory of Computing Library, 2016, no. 7, arXiv:1310.2035

[15] S. Aaronson and Y. Shi, "Quantum lower bounds for the collision and the element distinctness problems," *Journal of the ACM*, vol. 51, no. 4, pp. 595–605, Jul. 2004.

[16] A. Belovs and R. Špalek, "Adversary lower bound for the k-sum problem," in *Proceedings of the 4th Innovations in Theoretical Computer Science Conference (ITCS)*, 2013, pp. 323–328, arXiv:1206.6528

[17] M. Bun, R. Kothari, and J. Thaler, "The polynomial method strikes back: Tight quantum query bounds via dual polynomials," in *Proceedings of the 50th ACM Symposium on the Theory of Computing (STOC)*, 2018, pp. 297–310, arXiv:1710.09079

[18] M. Zhandry, "A note on the quantum collision and set equality problems," *Quantum Information and Computation*, vol. 15, no. 7&8, pp. 557–567, 2015, arXiv:1312.1027

[19] S. Aaronson and A. Ambainis, "Forrelation: A problem that optimally separates quantum from classical computing," in *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC)*, 2015, pp. 307–316, arXiv:1411.5729

[20] M. W. Liebeck, "On minimal degrees and base sizes of primitive permutation groups," *Archiv der Mathematik*, vol. 43, no. 1, pp. 11–15, 1984.

[21] B. Zhan, S. Kimmel, and A. Hassidim, "Super-polynomial quantum speed-ups for Boolean evaluation trees with hidden structure," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS)*, 2012, pp. 249–265, arXiv:1101.0796

[22] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, "Exponential algorithmic speedup by quantum walk," in *Proceedings of the 35th ACM Symposium on the Theory of Computing (STOC)*, 2003, pp. 59–68, arXiv:quant-ph/0209131

[23] A. Hulpke, "Notes on computational group theory," 2010, https://www.math.colostate.edu/~hulpke/CGT/cgtnotes.pdf.

[24] A. Kerber, *Applied finite group actions*, ser. Algorithms and Combinatorics. Springer Science & Business Media, 2013, vol. 19.

[25] S. Kimmel, "Quantum adversary (upper) bound," in *Proceedings of the 12th International Colloquium on Automata, Languages, and Programming (ICALP)*, ser. Lecture Notes in Computer Science, vol. 7391, 2012, pp. 557–568, arXiv:1101.0797