Efficient Quantum Algorithms for (Gapped) Group Testing and Junta Testing

Andris Ambainis* Aleksandrs Belovs† Oded Regev‡ Ronald de Wolf§

Abstract

In the k-junta testing problem, a tester has to efficiently decide whether a given function $f : \{0,1\}^n \to \{0,1\}$ is a k-junta (i.e., depends on at most k of its input bits) or is ε-far from any k-junta. Our main result is a quantum algorithm for this problem with query complexity $\widetilde{O}(\sqrt{k/\varepsilon})$ and time complexity $\widetilde{O}(n\sqrt{k/\varepsilon})$. This quadratically improves over the query complexity of the previous best quantum junta tester, due to Atıcı and Servedio. Our tester is based on a new quantum algorithm for a gapped version of the combinatorial group testing problem, with an up to quartic improvement over the query complexity of the best classical algorithm. For our upper bound on the *time* complexity we give a near-linear time implementation of a shallow variant of the quantum Fourier transform over the symmetric group, similar to the Schur-Weyl transform. We also prove a lower bound of $\Omega(k^{1/3})$ queries for junta-testing (for constant ε).

*Faculty of Computing, University of Latvia. Supported by the European Commission FET-Proactive project QALGO, ERC Advanced Grant MQC and Latvian State Research programme NexIT project No.1.

†CWI, the Netherlands. Supported by European Commission FET-Proactive project Quantum Algorithms (QALGO) 600700. Part of this work was done while at CSAIL, Massachusetts Institute of Technology, USA, supported by Scott Aaronson's Alan T. Waterman Award from the National Science Foundation, and at Faculty of Computing, University of Latvia.

‡Courant Institute of Mathematical Sciences, New York University. Supported by the Simons Collaboration on Algorithms and Geometry and by the National Science Foundation (NSF) under Grant No. CCF-1320188. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

§CWI and University of Amsterdam, the Netherlands. Partially supported by ERC Consolidator Grant QPROGRESS and by the European Commission FET-Proactive project Quantum Algorithms (QALGO) 600700.

1 Introduction.

1.1 Quantum property testing. Many computational problems are too hard to solve perfectly in any reasonable amount of time (especially if $\mathsf{P} \neq \mathsf{NP}$, as seems likely). Accordingly, much of theoretical as well as practical computer science is about trying to efficiently solve those problems in a weaker sense. Examples are trying to *approximate* the optimal solution, trying to solve the problem fast *on average*, trying to solve it fast *on most instances*, etc. A structured model for the latter is *property testing*. Here our goal is to test whether a given (usually very large) object f has a certain property $\mathcal{P}$. Typically the hardest instances of the problem are the ones that are on the boundary, "just outside" of the property, where one needs to look at a large part of f to decide if it is in or out of the property. But, in many cases such instances tend to appear due to noise or other imperfections, and as such should not really be rejected. The setting of property testing excludes such instances: it assumes that the given instance f either has the property $\mathcal{P}$, or is at least somewhat "far" from $\mathcal{P}$ (i.e., far from all instances that have property $\mathcal{P}$, according to some suitable distance measure). This "promise" on the inputs makes many hard problems much easier, and many property testers have been found over the last two decades to efficiently test properties of very large objects, see for instance [32]. Note that a tester even allows us to conclude something about inputs that are outside of the promise: if a tester accepts input f with high probability, then f must be close to at least one element that has the property $\mathcal{P}$.

In this paper we focus on *quantum* algorithms for property testing. These are substantially less studied than classical algorithms, but quantum property testing has been receiving increasing attention in the last few years, both for testing properties of classical objects and for testing properties of quantum objects. See [39] for a recent survey.

1.2 Group testing. We first develop a new quantum algorithm for a version of the (combinatorial) *group*

testing problem.[1] Group testing was invented in World War II to efficiently identify ill soldiers [28]. Suppose n soldiers have each given samples of their blood, and up to k of them are ill. One way to identify the ill ones is to separately test each of the n blood samples. However, blood tests are expensive, and if $k \ll n$ then something much more efficient can be done. By combining parts of the blood samples of a subset S of all soldiers and testing the combined sample, we can determine whether *at least one* of the soldiers in S has the disease, at the expense of only one blood test. Using binary search we can then identify one ill soldier using $O(\log n)$ tests, and all k ill soldiers using $O(k \log n)$ tests.[2]

Here we consider a "gapped" decision version of the group testing problem, which in its simplest form is the following:[3]

> **Gapped group testing (GGT), informal.** For some set $A \subseteq [n]$, define $f_A \colon 2^{[n]} \to \{0,1\}$ by setting $f_A(S) = 1$ iff S intersects A. Given the ability to query an f_A where either $|A| \leq k$ or $|A| \geq k + d$, decide which is the case.

Note that the function f_A is like a blood test, where A is the set of all ill soldiers and the input S is the set of the soldiers whose blood we include in the tested sample. The function outputs 1 exactly when at least one of the soldiers in S is ill.

Recently, Belovs [9] showed that if $|A| \leq k$, then one can *identify* A with $O(\sqrt{k})$ quantum queries to f_A. Clearly this algorithm can be used to solve the GGT problem for $d = 1$. The randomized query complexity of this problem for $d = 1$ is $\widetilde{\Theta}(k)$ (see Section 3.1 for references and proofs about the randomized case), so we have a quadratic quantum improvement over classical.

Things get more interesting as d grows. Using a simple modification of the algorithm in [9], in Section 3 we construct an optimal $\Theta(1 + \sqrt{k/d})$-query quantum algorithm for the GGT problem. The randomized complexity of the same problem is $\widetilde{\Theta}(k)$ for $d \leq \sqrt{k}$ and $\widetilde{\Theta}(1 + (k/d)^2)$ for $d \geq \sqrt{k}$.

The algorithm is constructed from a feasible solution to the (dual) semidefinite program for the *adversary bound*. Reichardt *et al.* [34, 41, 38] have shown that the adversary bound characterizes quantum query complexity up to a constant factor. This means that, in principle, any quantum query algorithm can be derived as a solution to this semidefinite program. However, the number of new quantum algorithms that have actually been obtained in this way remains fairly small. Moreover, the majority of these algorithms are developed in the *learning graph* methodology [7], while our algorithm is based on different ideas.

We subsequently use this quantum algorithm as a subroutine for our junta testing algorithm, but we feel the GGT problem is quite interesting by itself as well, and may find applications elsewhere. We now mention several other reasons why our GGT algorithm is interesting.

Fourth-power improvement. By considering our bounds on the complexity of the GGT problem, we see that there is a *quartic* (fourth-power) quantum improvement in query complexity for the regime $\sqrt{k} \leq d \leq k$. Most speed-ups obtained by quantum algorithms are either exponential (mostly, for computational problems from algebra or number theory) or quadratic or less (for algorithms based on Grover's algorithm or its generalizations). In contrast, our algorithm provides a fourth-power speedup, which is quite surprising given that it is based on the OR function, for which the best-possible quantum speedup is only quadratic. Only a few other examples of such speed-ups are known [46, 35, 11, 9].

Very recently, several new separations were found for *total* Boolean functions: a quartic speed-up of bounded-error quantum algorithms over *deterministic* classical algorithms [3], and a 2.5th-power speed-up of bounded-error quantum algorithms over bounded-error classical algorithms [14].

Robustness. Our algorithm still works if the action of the input function $f_A(S)$ is not defined for some values of S. For instance, if $|A| = k$ and S intersects A, the value of $f_A(S)$ can be anything: 0, 1, or even undefined. The same is true if $|A| = k + d$ and $S \cap A = \emptyset$. We say that such S are *irrelevant variables*. This property turns out to be very useful in applications of the GGT algorithm, in particular when we use it as a subroutine in our junta tester.

Time-efficient implementation. Our algorithm is one of the few quantum algorithms derived from the adversary bound with a *time-efficient* implementation, i.e., one that is efficient in total number of gates as well as in total number of queries (in general, the time complexity of the adversary-derived algorithm can be exponentially large in the number of input bits to the problem). Other examples are the formula-evaluation algorithm of Reichardt and Špalek [43] and the algorithm

[1] To avoid confusion: while we use the established term "group testing" here, this is *not* a property testing problem in the sense described above, because we have no "in the property or far from the property" promise here.

[2] $O(k \log n)$ tests is essentially optimal for simple information-theoretic reasons: each test gives at most one bit of information, but by identifying the k ill soldiers we learn $\log \binom{n}{k} = \Omega(k \log(n/k))$ bits of information. There is a large literature that optimizes the constant factor and other aspects of group testing algorithms, see for instance [29].

[3] The actual problem is slightly more general, see Definition 3.2.

for *st*-connectivity of Belovs and Reichardt [12].

The time complexity of our algorithm is $\widetilde{O}(n\sqrt{k/d})$, roughly n times its query complexity. This is probably the best one can hope for: the oracle takes an n-qubit input register, so it takes $\Omega(n)$ gates just to touch all those qubits.

The key to our time-efficient algorithm is an efficient, $\widetilde{O}(n)$-time, implementation of the quantum Fourier transform (QFT) on the linear space which we denote by M^n. It is of dimension 2^n and has an orthonormal basis indexed by the set of all subsets of $[n]$. The symmetric group $\mathbb{S}_n$ acts naturally on this space by permuting its basis elements. Our implementation, is close to the efficient quantum Schur-Weyl transform of Bacon, Chuang and Harrow [5, 6]. To the best of our knowledge, this is the first "algorithmic" application of this transformation ([6] lists a number of applications of this transformation for quantum protocols).

1.3 Junta testing. Our main result is about *junta testing*. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and $J \subseteq [n]$ be the set of (indices of) variables on which the function depends. We say that f is a *k-junta* if $|J| \leq k$. Such functions are often studied, for instance in learning theory if most of the features are irrelevant for the concept that needs to be learned (e.g., in biology if only a few genes determine some biological property). We say that f is *ε-far from any k-junta* if the normalized Hamming distance between f and g is at least ε for every k-junta g (i.e., f and g differ on at least $\varepsilon 2^n$ inputs). The *k-junta testing problem* is:

> **k-junta testing.** Given the ability to query an $f\colon \{0,1\}^n \to \{0,1\}$ that is either a k-junta or ε-far from any k-junta, decide which is the case.

We would like to test this efficiently. The primary measure of efficiency is the number of "queries," evaluations of f, which are usually the most expensive part of an algorithm. However, we will also consider time complexity later.

Junta testing has been well-studied in the last decade, see [16] for a recent survey. Classically, the best known tester is by Blais [15] and uses $O(k \log k + k/\varepsilon)$ queries to f, quadratically improving upon an earlier tester of [30]. The best known classical *lower* bound is $\Omega(k)$ for constant ε [25].

The best *quantum* tester, due to Atıcı and Servedio [4], uses $O(k/\varepsilon)$ queries. It is based on Fourier sampling. This quantum tester is better than Blais's classical tester by a $\log k$-factor (for constant ε), but does not beat the best known classical lower bound, leaving open the possibility of an equally efficient classical tester.

Our main result in this paper is a quantum tester with query complexity $\widetilde{O}(\sqrt{k/\varepsilon})$, which (up to logarithmic factors) quadratically improves over the previous best quantum junta tester and actually beats the known classical lower bound for the first time. We also give a time-efficient implementation.

Main theorem (informal). There is a quantum k-junta tester that uses $\widetilde{O}(\sqrt{k/\varepsilon})$ queries and $\widetilde{O}(n\sqrt{k/\varepsilon})$ time (i.e., elementary quantum gates and query gates).

Similarly to the GGT problem, this time complexity is the best that one could reasonably expect given our query complexity, because each query to f involves an n-qubit input register.

Our junta tester is described in Section 4. The idea is the following (suppressing the dependence on ε for simplicity). If f is far from any k-junta, then it depends on some $K > k$ variables, and together those $K - k$ "extra" variables will have at least ε "influence" (this will be quantified using the Fourier coefficients of f). We use the GGT algorithm to distinguish this case from the case when the function depends only on k variables.[4] The larger K is, the simpler it is to solve the GGT problem, but the harder it is to detect influential variables, as the ε influence gets spread over more variables. These two effects cancel each other out, and it requires $\widetilde{O}(\sqrt{k})$ many queries to distinguish the two cases, independently of the value of K.

Let us now briefly mention the lower bounds we obtain. As already noted by Atıcı and Servedio [4] and explained in Section 6, the classical lower bound approach for junta testing fails for quantum algorithms, because the corresponding instances can be easily solved quantumly in $O(\log k)$ queries. Instead of this, in Section 6 we describe a different approach using reduction from the problem of testing image size of a function. This already gives a lower bound of $\Omega(k^{1/3})$ by the Aaronson-Shi lower bound for the collision problem [1]. We believe that the actual complexity of testing support size of a distribution is around $\Omega(\sqrt{k})$, but proving this seems to require techniques beyond the state of the art in quantum lower bounds.

2 Preliminaries.

We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$, and 2^A to denote the set of subsets of A. A k-subset is a subset of size k. All matrices in this paper have real entries. If A is a matrix, $A[\![i, j]\!]$ denotes the element at row i and column j. A *projector* always stands for an orthogonal

[4] In the setting of classical testers, García-Soriano [31, p. 111] also noted "a striking resemblance between group testing and junta testing."

projector. We use Π_S to denote the projector onto a subspace S. We use log and ln to denote logarithms in base 2 and e, respectively. The notation $\mathbb{F}_q$ stands for a finite field with q elements.

We assume familiarity with basic probability theory. Let $\mathcal{B}(k,p)$ denote the binomial distribution: $\Pr[\mathcal{B}(k,p)=i]=\binom{k}{i}p^i(1-p)^{k-i}$. We use $\mathcal{H}_n(k,m)$ to denote the hypergeometric distribution, i.e., the distribution of $\big|A\cap[k]\big|$ when A is sampled from all m-subsets of $[n]$ uniformly at random. By $X\sim\mathcal{B}$, we denote that X is sampled from probability distribution $\mathcal{B}$.

2.1 Quantum algorithms. Let us define quantum query algorithms. For a more complete treatment see [21]. A quantum query algorithm is defined as a sequence of unitary transformations alternating with oracle calls:

(2.1)
$$U_0 \to O_x \to U_1 \to O_x \to \cdots \to U_{T-1} \to O_x \to U_T.$$

Here the U_is are arbitrary unitary transformations that are independent of the input. The input oracle O_x is the same throughout the algorithm, and is the only way the algorithm accesses the input string $x=(x_j)$. The input oracle decomposes in the following way:

$$O_x = \bigoplus\nolimits_{j\in[n]} O_{x,j}, \tag{2.2}$$

where $O_{x,j}$ is some unitary transformation that only depends on the symbol x_j. In this paper x will be a Boolean string, and we adopt the following convention: $O_{x,j}=I$ if $x_j=0$, and $O_{x,j}=-I$ if $x_j=1$, where I is the identity operator.

The computation starts in a predefined state $|0\rangle$. After all the operations in (2.1) are performed, some predefined output register is measured. We say that the algorithm *computes* a function F (with bounded error) if, for any x in the domain, the result of the measurement is $F(x)$ with probability at least $2/3$. The number T is the *query complexity* of the algorithm. The smallest value of T among all algorithms computing f is the quantum query complexity of F, and is denoted by $Q(F)$.

We will also be interested in *time complexity* (also known as *gate complexity*) of the algorithm. It is defined as the total number of elementary quantum gates (from some fixed universal set of gates) required to implement all the unitary transformations $U_0,\ldots,U_T$.

One of our main algorithmic tools is amplitude amplification. This is encapsulated in the following result of Brassard *et al.* [19, Section 2], which generalizes Grover's quantum search algorithm [33].

LEMMA 2.1. (AMPLITUDE AMPLIFICATION) *Let $\mathcal{A}$ be some quantum procedure and S some set of basis states on the algorithm's output space. Suppose that the probability that measuring the state $\mathcal{A}|0\rangle$ gives a basis state in S is at least p. Then there exists another procedure $\mathcal{B}$, which invokes $\mathcal{A}$ and $\mathcal{A}^{-1}$ $O(1/\sqrt{p})$ many times (we sometimes call such an invocation a "round of amplitude amplification"), such that the probability that measuring the state $\mathcal{B}|0\rangle$ gives a basis state in S is at least $9/10$. If, in contrast, the probability of obtaining a basis state in S when measuring $\mathcal{A}|0\rangle$ was 0, this probability will still be 0 when measuring $\mathcal{B}|0\rangle$.*

For time-efficient implementation of our algorithm, we need the following two results.

THEOREM 2.2. (PHASE ESTIMATION [36, 26]) *Assume a unitary U is given as a black box. There exists a quantum algorithm that, given an eigenvector $|psi\rangle$ of U with eigenvalue $\mathrm{e}^{\mathrm{i}\phi}$, outputs a real number w such that $|w-\phi|\le\delta$ with probability at least $9/10$. Moreover, the algorithm uses $O(1/\delta)$ controlled applications of U and U^{-1} and $\frac{1}{\delta}\operatorname{polylog}(1/\delta)$ other elementary operations.*

LEMMA 2.3. (EFFECTIVE SPECTRAL GAP LEMMA [38]) *Let Π_1 and Π_2 be two orthogonal projectors in the same vector space (not necessarily* pairwise *orthogonal), and $R_1=2\Pi_1-I$ and $R_2=2\Pi_2-I$ be the reflections about their images. For $\delta\ge 0$, let P_δ be the projector on the span of all eigenvectors of R_2R_1 that have eigenvalues $\mathrm{e}^{\mathrm{i}\theta}$ with $|\theta|\le\delta$. Then, for any vector w in the kernel of Π_1, we have*

$$\|P_\delta\Pi_2 w\| \le \frac{\delta}{2}\|w\|.$$

2.2 Adversary bound. Here we describe the dual adversary bound, the main tool for the construction of our algorithms.

Let $F\colon\mathcal{D}\to\{0,1\}$, with $\mathcal{D}\subseteq\{0,1\}^n$, be a partial Boolean function. The (dual) adversary bound, $\mathrm{ADV}^{\pm}(F)$, is defined as the optimal value of the following semi-definite optimization problem:

(2.3a)
$$\text{minimize} \quad \max_{z\in\mathcal{D}} \textstyle\sum_{j\in[n]} X_j[\![z,z]\!]$$

(2.3b)
$$\text{s.t.} \quad \sum\nolimits_{j:x_j\neq y_j} X_j[\![x,y]\!] = 1 \quad \text{for all } F(x)\neq F(y);$$

(2.3c)
$$X_j \succeq 0 \qquad \text{for all } j\in[n],$$

where X_j are $\mathcal{D}\times\mathcal{D}$ positive semi-definite matrices. Recall that $Q(F)$ denotes the bounded-error quantum query complexity of F. Then, we have the following important result.

THEOREM 2.4. ([34, 42, 38]) *For every F, $Q(F) = \Theta(\mathrm{ADV}^{\pm}(F))$.*

Because of Theorem 2.4, one may come up with a solution to the adversary bound instead of explicitly constructing a quantum algorithm. This is how we construct the algorithm in Section 3. The following "unweighted adversary bound" is a useful special case (and precursor) of the general adversary lower bound:

THEOREM 2.5. ([2]) *Suppose there is a non-empty relation $R \subseteq F^{-1}(1) \times F^{-1}(0)$ that satisfies*

(i) for each $x \in F^{-1}(1)$ appearing in R, there are at least m distinct $y \in F^{-1}(0)$ such that $(x,y) \in R$;

(ii) for each $y \in F^{-1}(0)$ appearing in R, there are at least m' distinct $x \in F^{-1}(1)$ such that $(x,y) \in R$;

(iii) for each $x \in F^{-1}(1)$ and each $j \in [n]$, there are at most ℓ distinct $y \in F^{-1}(0)$ such that $(x,y) \in R$ and $x_j \neq y_j$;

(iv) for each $y \in F^{-1}(0)$ and each $j \in [n]$, there are at most ℓ' distinct $x \in F^{-1}(1)$ such that $(x,y) \in R$ and $x_j \neq y_j$;

Then, the bounded-error quantum query complexity of F is $\Omega\left(\sqrt{\frac{mm'}{\ell\ell'}}\right)$.

The adversary bound is also useful for function composition. Assume $F\colon \mathcal{D} \to \{0,1\}$, with $\mathcal{D} \subseteq \{0,1\}^n$, and, for any $j \in [n]$, let G_j be a partial Boolean function on m_j variables. The composed Boolean function $F \circ (G_1, \ldots, G_n)$ on $\sum_{j=1}^n m_j$ variables is defined by

$$(x_{11}, \ldots, x_{1m_1}, \ldots, x_{n1}, \ldots, x_{nm_n}) \mapsto F\big(G_1(x_{11}, \ldots, x_{1m_1}), \ldots, G_n(x_{n1}, \ldots, x_{nm_n})\big), \tag{2.4}$$

where the composed function is defined on the input $(x_{11}, \ldots, x_{nm_n})$ iff the values of all G_j on the right-hand side of (2.4) are defined, and the corresponding n-tuple belongs to $\mathcal{D}$.

THEOREM 2.6. ([41]) *We have*

$$\mathrm{ADV}^{\pm}\big(F \circ (G_1, \ldots, G_n)\big) \leq \mathrm{ADV}^{\pm}(F) \max_{j \in [n]} \mathrm{ADV}^{\pm}(G_j).$$

In particular, this theorem together with Theorem 2.4 implies

COROLLARY 2.7. (TIGHT COMPOSITION RESULT)
$$Q\big(F \circ (G_1, \ldots, G_n)\big) = O\big(Q(F) \max_{j \in [n]} Q(G_j)\big).$$

That is, one can compose functions without the logarithmic overhead in query complexity that arises in the standard method of composition (which would reduce the error probability of the algorithm for the internal functions to $\ll 1/n$ by taking the majority-outcome of $O(\log n)$ independent runs of the algorithm).

2.3 Irrelevant Variables. Consider the following motivating example. Let $G_1, \ldots, G_n$ be partial Boolean functions. We define the "robust conjunction" of the functions G_i,

$$H(x) = \widetilde{\bigwedge}_{i \in [n]} G_i(x)\ , \tag{2.5}$$

as the partial Boolean function H given by the following: if $G_1(x) = \cdots = G_n(x) = 1$ (in particular, x is in the domain of all G_i), then $H(x) = 1$; if there exists an i for which $G_i(x) = 0$ (in particular, x is in the domain of G_i), then $H(x) = 0$; and otherwise $H(x)$ is not defined. Note that in the second case x may lie outside the domain of G_j for some (or even all) $j \neq i$.

One interpretation of this expression, which we use in Section 4, is as follows. The function H is some test for x, and the G_i are sub-tests, which check for different possibilities of how H can fail. Thus, a positive input must satisfy all the sub-tests, whereas a negative x has to fail at least one sub-test G_i but might give an ambiguous answer on other tests G_j.

Classically, the above is a non-issue, since we can always apply the algorithm for G_i on an input x, even if that input is outside the domain of G_i—the algorithm's output must still be either 0 or 1. Quantumly, the situation is more delicate: strictly speaking, we cannot apply the textbook Grover search to evaluate H since the oracle in the definition of a quantum query algorithm is supposed to apply either I or $-I$ on each input, but a quantum algorithm for G_i on an input x outside its domain may apply an arbitrary unitary transformation on its entire working space.

In this particular case, H can be evaluated using amplitude amplification, Lemma 2.1, instead of the usual Grover search. In Proposition 2.11 below, we extend this result to the case when the conjunction is replaced by an arbitrary partial Boolean function F. Additionally, we show how to generalize the tight composition result, Corollary 2.7, to this more general setting. In order to do this, we have to make a number of definitions.

DEFINITION 2.8. (IRRELEVANT VARIABLES) *Let $F\colon \mathcal{D} \to \{0,1\}$ be a partial Boolean function with the domain $\mathcal{D} \subseteq \{0,1\}^n$. For each input $x \in \mathcal{D}$, some input variables $j \in [n]$ may be called* irrelevant*, the remaining variables called* relevant*. This can be done in an arbitrary way, as long as the following* consistency condition *is satisfied: for any $x, y \in \mathcal{D}$ such that $F(x) \neq F(y)$, there must exist a variable j relevant to both x and y and such that $x_j \neq y_j$.*

DEFINITION 2.9. (EVALUATION) *Evaluation of the function F with irrelevant variables is defined as in*

Section 2.1, with the difference that, for an input $z \in \mathcal{D}$, the input oracle may malfunction on irrelevant variables, i.e., $O_{z,j}$ in (2.2) may be an arbitrary unitary if j is irrelevant for z.

DEFINITION 2.10. (COMPOSITION) *The composition $F \circ (G_1, \ldots, G_n)$ with irrelevant variables is defined as in (2.4) but on a larger domain. Namely, the right-hand side of (2.4) is defined iff there exists $z \in \mathcal{D}$ such that $z_j = G_j(x_{j1}, \ldots, x_{jm_j})$ for all relevant j. In particular, the value of $G_j(x_{j1}, \ldots, x_{jm_j})$ need not be defined for irrelevant j. The value of the composed function on this input is then set to $F(z)$, and does not depend on the particular choice of z.*

We use the last two definitions as follows. Definition 2.10 is used in Section 4 to get a *query-efficient* algorithm for testing juntas, using Corollary 2.12 below. Thus we save a logarithmic factor, as described after Corollary 2.7. Definition 2.9 is used in Section 5 to get a *time-efficient* implementation of the algorithm from Section 4 (again, see the discussion after Corollary 2.7).

The following proposition is a special case of the construction in [10].

PROPOSITION 2.11. *Let (X_j) be a feasible solution to the adversary bound (2.3) with objective value T. Call an input variable j is* irrelevant *for an input $z \in \mathcal{D}$ iff $X_j[\![z, z]\!] = 0$. With this choice of irrelevant variables,*

(a) There exists a quantum algorithm that evaluates the function F in the sense of Definition 2.9, using $O(T)$ queries.

(b) For arbitrary partial Boolean functions G_j, we have

$$\mathrm{ADV}^{\pm}\big(F \circ (G_1, \ldots, G_n)\big) \le T \max_{j \in [n]} \mathrm{ADV}^{\pm}(G_j),$$

where $F \circ (G_1, \ldots, G_n)$ is as in Definition 2.10.

It is easy to see that this choice of irrelevant variables satisfies the consistency condition of Definition 2.8. Indeed, if $F(x) \neq F(y)$, then (2.3b) implies the existence of j with $X_j[\![x, y]\!] \neq 0$, and since $X_j \succeq 0$, both $X_j[\![x, x]\!]$ and $X_j[\![y, y]\!]$ are non-zero.

The proof of point (a) is analogous to the proof of the upper bound of Theorem 2.4, and we will skip the details here. However, we will prove in Section 5.1 that our time-efficient implementation of the corresponding solution to the adversary bound has this property.

See the full version of the paper for a self-contained proof of point (b). It immediately gives the following variant of Corollary 2.7.

COROLLARY 2.12. *Let (X_j) be a feasible solution to the adversary bound (2.3) with objective value T. We say that an input variable j is irrelevant for an input $z \in \mathcal{D}$ iff $X_j[\![z, z]\!] = 0$. For arbitrary partial Boolean functions G_j, we have*

$$Q\big(F \circ (G_1, \ldots, G_n)\big) = O\big(T \max_{j \in [n]} Q(G_j)\big),$$

with the composition as in Definition 2.10 with this choice of irrelevant variables.

EXAMPLE 2.13. (AND) *Let us return to the example in (2.5). Consider the AND function on the domain $\mathcal{D} = \{z \in \{0,1\}^n \mid |z| \ge n-1\}$, where $|z|$ is the Hamming weight. A feasible solution to (2.3) for this function is given by $X_j = \psi_j \psi_j^*$, where $\psi_j \in \mathbb{R}^{\mathcal{D}}$ is given by*

$$\psi_j[\![z]\!] = \begin{cases} n^{-1/4}, & \text{if } |z| = n; \\ n^{1/4}, & \text{if } |z| = n-1 \text{ and } z_j = 0; \\ 0, & \text{if } |z| = n-1 \text{ and } z_j = 1. \end{cases}$$

The objective value of this solution is $\sqrt{n}$. Note that if $|z| = n-1$, then any variable j with $z_j = 1$ is irrelevant for this input. This coincides with our definition of the "robust conjunction" at the beginning of this section.

2.4 Fourier analysis. We use Fourier analysis for arbitrary real-valued functions $f\colon \{0,1\}^n \to \mathbb{R}$. If f is Boolean, it is usually convenient to assume that its range is $\{\pm 1\} = \{1, -1\}$ rather than $\{0,1\}$. For a string $s \in \{0,1\}^n$, the corresponding *character* is a Boolean function $\chi_s\colon \{0,1\}^n \to \{\pm 1\}$ defined by $\chi_s(x) = (-1)^{s \cdot x}$, where $s \cdot x = \sum_j s_j x_j$ denotes the inner product of s and x. We will often use the corresponding subset $S \subseteq [n]$ instead of a string $s \in \{0,1\}^n$.

Every function $f\colon \{0,1\}^n \to \mathbb{R}$, has a *Fourier decomposition* as follows:

$$f(x) = \sum_{s \in \{0,1\}^n} \widehat{f}(s) \chi_s(x),$$

where $\widehat{f}(s) = 2^{-n} \sum_x f(x) \chi_s(x)$ is the *Fourier coefficient*. The set $\big\{s \mid \widehat{f}(s) \neq 0\big\}$ is called the *(Fourier) spectrum* of f. *Parseval's identity* says that rhis transformation respects the norm: $\mathbb{E}_x\big[f(x)^2\big] = \sum_s \widehat{f}(s)^2$. In particular, for a Boolean $f\colon \{0,1\}^n \to \{\pm 1\}$, we have $\sum_s \widehat{f}(s)^2 = 1$.

For a subset $S \subseteq [n]$, we define the *influence* of S on f by

$$\mathrm{Inf}_S(f) = \sum_{T\colon T \cap S \neq \emptyset} \widehat{f}(T)^2. \tag{2.6}$$

If S consists of a single element $j \in [n]$ we write $\mathrm{Inf}_j(f)$, which is $\sum_{T\colon j \in T} \widehat{f}(T)^2$.

An alternative (but equivalent) definition of influence for functions with range $\{\pm 1\}$ is as follows. Consider the following randomized procedure. Generate $x \in \{0,1\}^n$ uniformly at random. Obtain $y \in \{0,1\}^n$ from x by replacing, for each $j \in S$, x_j by an independent uniformly random bit. Then, the influence $\mathrm{Inf}_S(f)$ is precisely twice the probability that $f(x) \neq f(y)$. Note that the influence $\mathrm{Inf}_j(f)$ equals the probability that $f(x) \neq f(x^{\oplus j})$ when x is sampled from $\{0,1\}^n$ uniformly at random, where $x^{\oplus j}$ denotes x with the jth bit flipped. We repeatedly use the following two obvious properties of influence:

- Monotonicity. If $S \subseteq T$, then $\mathrm{Inf}_S(f) \leq \mathrm{Inf}_T(f)$.
- Subadditivity. $\mathrm{Inf}_{S \cup T}(f) \leq \mathrm{Inf}_S(f) + \mathrm{Inf}_T(f)$ for all $S, T \subseteq [n]$.

The following lemma from [30, 4] explains why influence is important in our junta testing algorithm.

LEMMA 2.14. *If f is ε-far from any k-junta, then for all $W \subseteq [n]$ of size $|W| \leq k$ we have $\mathrm{Inf}_{[n] \setminus W}(f) \geq \varepsilon$.*

Proof. Define a (not necessarily Boolean) function $g\colon \{0,1\}^n \to \mathbb{R}$ by $g(x) = \sum_{S \subseteq W} \widehat{f}(S)\chi_S(x)$. Let h be the Boolean function that is the sign of g. This h only depends on the variables in W, so it is a k-junta. Since f is ε-far from any k-junta, we have (using Parseval's identity)

$$\varepsilon \leq \frac{\big|\{x \mid f(x) \neq h(x)\}\big|}{2^n} \leq \mathop{\mathbb{E}}_x\big[(f(x) - g(x))^2\big] = \sum_S (\widehat{f}(S) - \widehat{g}(S))^2 = \sum_{S \not\subseteq W} \widehat{f}(S)^2.$$

3 Gap version of group testing.

Our junta testers, which we describe in the next section, work by generating a random subset $S \subseteq [n]$ and testing whether it intersects the set of influential variables of f. In this section we study a more abstract problem, where we assume that we have an oracle that answers this intersection-question with certainty. We believe this problem is of independent interest.

For each $A \subseteq [n]$, define the function $\mathrm{Intersects}_A\colon 2^{[n]} \to \{0,1\}$ by

$$\mathrm{Intersects}_A(S) = \begin{cases} 1, & \text{if } A \cap S \neq \emptyset; \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

In the standard version of group testing [29] one is given oracle access to the function $\mathrm{Intersects}_A$ for some A with $|A| \leq k$, and the task is to identify A. A natural variant of this problem is to compute or approximate the cardinality of A (a task already considered in the group testing literature [27]), and a decision version of the latter is deciding whether that cardinality is k or $k + d$ for some $d, k \geq 1$. We define this formally next.

DEFINITION 3.1. (EGGT) *Let k and d be positive integers, $\mathcal{X}$ consist of all subsets of $[n]$ having size exactly k, and $\mathcal{Y}$ consist of all subsets of $[n]$ having size exactly $k + d$. In the* exact gap version of the group testing (EGGT) problem *with parameters k and d, one is given oracle access to the function $\mathrm{Intersects}_A$ with $A \in \mathcal{X} \cup \mathcal{Y}$, and the task is to decide whether $A \in \mathcal{X}$ or $A \in \mathcal{Y}$.*

We also study a relaxation of EGGT, in which we allow "false negatives" in the small-set case and "false positives" in the large-set case. This will be convenient for our applications. Luckily, algorithms for solving EGGT often turn out to also solve this harder problem.

DEFINITION 3.2. (GGT) *Let k and d be positive integers. Define two families of functions*

$$\widetilde{\mathcal{X}} = \Big\{ f\colon 2^{[n]} \to \{0,1\} \;\Big|\; \exists A \in \mathcal{X}\ \forall S \subseteq [n] : S \cap A = \emptyset \implies f(S) = 0 \Big\} \tag{3.8}$$

and

$$\widetilde{\mathcal{Y}} = \Big\{ f\colon 2^{[n]} \to \{0,1\} \;\Big|\; \exists B \in \mathcal{Y}\ \forall S \subseteq [n] : S \cap B \neq \emptyset \implies f(S) = 1 \Big\}. \tag{3.9}$$

In the gap version of the group testing (GGT) problem *with parameters k and d, one is given oracle access to $f \in \widetilde{\mathcal{X}} \cup \widetilde{\mathcal{Y}}$, and the task is to decide whether $f \in \widetilde{\mathcal{X}}$ or $f \in \widetilde{\mathcal{Y}}$.*

It is easy to see that EGGT is a special case of the GGT, where the implications in (3.8) and (3.9) are replaced by equivalences. The GGT problem also includes as a special case the problem of distinguishing a function $\mathrm{Intersects}_A(f)$ with $|A| \leq k$ from a function $\mathrm{Intersects}_A(f)$ with $|A| \geq k + d$.

3.1 Randomized complexity.

In this section we show that the randomized query complexity of the gap version of group testing is $\widetilde{\Theta}\big(\min\{k, 1 + (k/d)^2\}\big)$.

The upper bound already follows from the existing literature on group testing: an $O\big(1 + (k/d)^2\big)$ bound appears in [24] (with a surprisingly elaborate analysis) and an $O\big(k \log k\big)$ bound appears in [23] and independently in [31, Section 5.3]. Strictly speaking those results only apply to the EGGT problem. Because of that, and also for completeness, we include in Theorem 3.3 an upper bound for the more general GGT problem.

The only *lower* bound for randomized complexity that we are aware of is $\Omega(k)$ for the special case $d = 1$ due to García-Soriano [31, Section 5.3], who calls the problem "relaxed group testing." In Theorem 3.4, we give a short reduction from the Gap Hamming Distance problem in communication complexity which applies only to GGT. The full version of the paper contains a substantially longer but self-contained proof which applies to EGGT as well.

THEOREM 3.3. *For any $k, d \geq 1$, the randomized query complexity of the GGT problem with parameters k and d is $O\big(\min\{k \log k, 1 + (k/d)^2\}\big)$.*

Proof. We start with the easier upper bound of $O\big(1 + (k/d)^2\big)$. Take $S \subseteq [n]$ by including each element independently at random with probability $1/k$. If we are in the "small" case of GGT (as in (3.8)), the probability of $f(S) = 1$ is at most $1 - (1 - 1/k)^k$. If, on the other hand, we are in the "large" case of GGT (as in (3.9)) then that probability is at least $1 - (1 - 1/k)^{k+d}$. As these two probabilities differ by $\Omega(\min\{1, d/k\})$, by a Chernoff bound, we can distinguish the two cases by repeating this procedure $O\big(1 + (k/d)^2\big)$ times.

We now prove the upper bound $O\big(k \log k\big)$. The algorithm maintains a partition of $[n]$, initially set to the trivial partition $\{[n]\}$. Each set in the partition can be either active or inactive, with the initial set $[n]$ being active. We maintain the invariant that for all sets S in the partition, $f(S) = 1$. (We can assume $f([n]) = 1$ as otherwise we are clearly in the "small" case.) At each step of the algorithm, we take an active set S in the partition and repeat the following $10 \log k$ times. We partition S into S_1 and S_2 by taking each element of S independently to be in either S_1 or S_2 with probability $1/2$. We then query f on both S_1 and S_2. If f returns 1 on both, then we replace S with S_1 and S_2, and this ends the loop for S. Otherwise, if we did not manage to split S after $10 \log k$ attempts, we declare S to be inactive and move on to another set in the partition. If at any point the partition contains at least $k + 1$ sets we stop and output "large." Otherwise, if all (at most k) sets are inactive, we stop and output "small."

Notice that after each step we either add a set to the partition or declare a set inactive. There can therefore be at most $2k$ steps, and since each step involves at most $10 \log k$ queries, the total number of queries is at most $20k \log k$. The correctness in the "small" case is immediate from our invariant: the only way for the algorithm to output "large" is if there are $k + 1$ sets in the partition on which f returns 1 but this cannot happen in the small case. So consider the "large" case as in (3.9) with some set B of size at least $k+1$. We claim that with high probability, all inactive sets intersect B in at most 1 element, and hence it cannot happen that there are at most k sets in the partition and all are inactive. To see why, notice that if a set S intersects B in at least two elements, then there is probability $1/2$ that when we split S into S_1 and S_2, these two elements would end up in a different set. In this case f must answer 1 on both S_1 and S_2 and S would be split. Therefore the probability for such an S to become inactive is at most $2^{-10 \log k} = k^{-10}$, which means that with high probability this bad event will never happen.

THEOREM 3.4. *For any $k, d \geq 1$, the randomized query complexity of the GGT problem with parameters k and d is $\widetilde{\Omega}\big(\min\{k, 1 + (k/d)^2\}\big)$.*

Proof. Following the general approach to classical testing lower bounds of Blais *et al.* [17], we will show a reduction from the Gap Hamming Distance (GHD) problem in communication complexity. In this problem there are two parties, Alice and Bob. Alice receives a bit string $x \in \{0,1\}^n$ and Bob receives a bit string $y \in \{0,1\}^n$. Their goal is to decide if the Hamming weight of $x \oplus y$ (i.e., the Hamming distance between x and y) is greater than $n/2 + g$ or at most $n/2 - g$ (and can behave arbitrarily for values in between). They are allowed the use of shared randomness and, for each input (x, y), need to output the correct answer with probability, say, at least $2/3$. It was shown in [22] (see also [45]) that for any $1 \leq g < n/2$, any protocol solving this problem must use at least $\Omega(\min\{n, n^2/g^2\})$ bits of communication (and this is tight).

Now let $k, d \geq 1$. It clearly suffices to prove the theorem for $d < k$, so assume that this is the case. The result now follows from the observation that any algorithm solving GGT with parameters k and d making $q \leq k$ queries implies a protocol for GHD with parameters $n = 2k$ and $g = d$ using $O(q \log k)$ bits of communication. In this protocol, Alice and Bob simulate the GGT algorithm given oracle access to Intersects_A where $A \subseteq [n]$ is the support of $x \oplus y$, using their shared randomness as coins to the algorithm. Whenever the algorithm performs a query S, Alice and Bob compute $\text{Intersects}_A(S)$ by running an Equality protocol with error probability less than $1/(10k)$ and communication $O(\log k)$ (see [37, Example 3.13]) to check if the restrictions of x and y to S are identical. It remains to notice that if the Hamming weight of $x \oplus y$ is at most $n/2 - g = k - d$, then $\text{Intersects}_A \in \widetilde{\mathcal{X}}$, whereas if it is greater than $n/2 + g = k + d$, then $\text{Intersects}_A \in \widetilde{\mathcal{Y}}$.

3.2 Quantum complexity. The aim of this section is to show that the quantum query complexity of the GGT problem is $\Theta\big(1 + \sqrt{k/d}\big)$. Thus, when $\sqrt{k} \leq d \leq k$, the quantum algorithm provides a quartic improvement

over the randomized one. We start with a lower bound for EGGT, which implies the same lower bound for GGT.

PROPOSITION 3.5. *The quantum query complexity of the EGGT problem with parameters k and d is $\Omega(1+\sqrt{k/d})$.*

Proof. Take $n = k + d$. In this case $\mathcal{Y}$ contains only one element, namely $[n]$, and the corresponding function takes value 1 on every S except $S = \emptyset$. Intuitively, one detects that $A \in \mathcal{X}$ by finding an $i \notin A$, so the problem becomes the unstructured search problem of size n, where the d elements of $[n] \setminus A$ are marked. Unstructured search requires $\Omega\big(\sqrt{n/d}\big)$ queries [18].

This intuitive argument can be made rigorous via the unweighted adversary lower bound (Theorem 2.5). We put the one input in $\mathcal{Y}$ in relation with all $\binom{n}{k}$ inputs in $\mathcal{X}$, so $m = 1$ and $m' = \binom{n}{k}$. For each fixed nonempty $S \subseteq [n]$, there are $\binom{n-|S|}{k}$ different $A \in \binom{[n]}{k}$ such that $A \cap S = \emptyset$ (these are the $A \in \mathcal{X}$ where a query to S returns value 0, showing that $A \neq [n]$). This number is maximized for $|S| = 1$, so $\ell' \leq \binom{n-1}{k}$ (and $\ell = 1$). The lower bound from Theorem 2.5 is

$$\Omega\left(\sqrt{\frac{mm'}{\ell\ell'}}\right) = \Omega\left(\sqrt{\binom{n}{k}\Big/\binom{n-1}{k}}\right) = \Omega\big(\sqrt{n/d}\big) = \Omega(1+\sqrt{k/d}).$$

The remaining part of this section is devoted to showing a matching upper bound on the quantum query complexity. In Section 5 we will show how to implement this algorithm time-efficiently.

THEOREM 3.6. *There exists a quantum algorithm that solves the GGT problem with parameters k and d using $O\big(\sqrt{1+k/d}\big)$ queries.*

Proof. We construct a feasible solution to the semidefinite program (2.3) for the *EGGT problem*. The adversary bound reads as follows:

(3.10a)
$$\text{minimize} \quad \max_{A\in\mathcal{X}\cup\mathcal{Y}} \textstyle\sum_{S\subseteq[n]} X_S[\![A,A]\!]$$

(3.10b)
$$\text{s.t.} \sum_{S\colon A\cap S=\emptyset \text{ xor } B\cap S=\emptyset} X_S[\![A,B]\!] = 1 \quad \forall A\in\mathcal{X},\, B\in\mathcal{Y};$$

(3.10c)
$$X_S \succeq 0 \qquad \text{for all } S\subseteq[n].$$

Moreover, our solution will be such that

$$X_S[\![A,A]\!] = 0 \quad \text{if} \quad \begin{array}{l} A\in\mathcal{X} \text{ and } S\cap A\neq\emptyset, \text{ or} \\ A\in\mathcal{Y} \text{ and } |S\cap A|\neq 1. \end{array} \tag{3.11}$$

This ensures that a feasible solution to (3.10) also gives a feasible solution to the adversary bound for the *GGT problem*. Indeed, for a function $f \in \widetilde{\mathcal{X}}$, we can choose $A_f \in \mathcal{X}$ that satisfies the existential quantifier in (3.8). Similarly, for $g \in \widetilde{\mathcal{Y}}$, we can select $A_g \in \mathcal{Y}$ that satisfies the existential quantifier in (3.9). A feasible solution for the GGT problem then consists of $(\widetilde{\mathcal{X}} \cup \widetilde{\mathcal{Y}}) \times (\widetilde{\mathcal{X}} \cup \widetilde{\mathcal{Y}})$ matrices $\widetilde{X}_S$ given by $\widetilde{X}_S[\![f,g]\!] = X_S[\![A_f, A_g]\!]$. Thus, $\widetilde{X}_S$ is the matrix X_S with many repeated rows and columns. A simple case analysis involving (3.10b) and (3.11) shows that this is a feasible solution for the GGT problem.

Our feasible solution to (3.10) is an adaptation of the solution given in [9] for the task of *finding* the subset A. It is possible to give a solution to (3.10) in the style of [9]. However, below we give a more direct construction resulting in matrices X_S of rank 1.

Clearly, we may assume that $n \geq k+d$. Let $S \subseteq [n]$, and $s = |S|$. If $s = 0$ or $s > n-k-d+1$, we define $X_S = 0$. If $1 \leq s \leq n-k-d+1$, we define $X_S = \psi\psi^*$, where ψ is a vector indexed by sets $A \in \mathcal{X} \cup \mathcal{Y}$ with entries

$$\psi[\![A]\!] = \begin{cases} \alpha_s, & \text{if } A\in\mathcal{X} \text{ and } A\cap S=\emptyset; \\ \beta_s, & \text{if } A\in\mathcal{Y} \text{ and } |A\cap S|=1; \\ 0, & \text{otherwise.} \end{cases} \tag{3.12}$$

Here α_s and β_s are some positive real numbers satisfying

$$\alpha_s\beta_s = \left((n-k)\binom{n-k-1}{s-1}\right)^{-1}. \tag{3.13}$$

The values of α_s and β_s depend on d and will be chosen later in order to minimize the value of the objective function (3.10a). From (3.12), it is easy to conclude that (3.11) indeed holds.

Ignoring repeated and zero entries, X_S is essentially a 2×2 block matrix of the form

$$X_S = \begin{pmatrix} \alpha_s^2 & \alpha_s\beta_s \\ \alpha_s\beta_s & \beta_s^2 \end{pmatrix}.$$

The proof now follows from the two claims below.

CLAIM 3.7. *The matrices X_S form a feasible solution to (3.10b) for the EGGT problem and any value of d.*

Proof. Fix $A \in \mathcal{X}$, $B \in \mathcal{Y}$, and let $\ell = |B \setminus A| \geq d$. Note that $X_S[\![A,B]\!] = 0$ if the condition on S in the sum in (3.10b) is not satisfied. Next,

$$\sum_{S\subseteq[n]} X_S[\![A,B]\!] = \frac{1}{n-k}\sum_{s=1}^{n-k-\ell+1} \frac{\ell\binom{n-k-\ell}{s-1}}{\binom{n-k-1}{s-1}} = \frac{\ell}{n-k}\,T(n-k-\ell, n-k-1)\,,$$

where, for non-negative integers $a \le b$, we define

$$T(a,b) = 1 + \frac{a}{b} + \cdots + \frac{a(a-1)(a-2)\cdots 1}{b(b-1)(b-2)\cdots(b-a+1)} .$$

Thus, to show that $\sum_{S\subseteq[n]} X_S[\![A,B]\!] = 1$ it remains to show that $T(a,b) = (b+1)/(b-a+1)$. This is easy to check by induction on a: the case $a = 0$ is trivial, and for the inductive step we have

$$T(a,b) = 1 + \frac{a}{b}T(a-1,b-1) = \frac{b+1}{b-a+1} .$$

CLAIM 3.8. *For each d, there exists a choice of α_s and β_s satisfying (3.13) such that the objective value (3.10a) is $O(\sqrt{1+k/d})$.*

Proof. Fix a positive integer $s \le n-k-d+1$. For all $A \in \mathcal{X}$ and $B \in \mathcal{Y}$, we have

$$\sum_{S\subseteq[n]\colon |S|=s} X_S[\![A,A]\!] = \binom{n-k}{s}\alpha_s^2 \tag{3.14}$$

and

$$\sum_{S\subseteq[n]\colon |S|=s} X_S[\![B,B]\!] = (k+d)\binom{n-k-d}{s-1}\beta_s^2. \tag{3.15}$$

We take α_s and β_s so that the values of (3.14) and (3.15) are equal. In particular, they are equal to their geometric mean, which, by (3.13), is

$$\frac{\sqrt{\binom{n-k}{s}(k+d)\binom{n-k-d}{s-1}}}{(n-k)\binom{n-k-1}{s-1}} \le \sqrt{\frac{k+d}{s(n-k)}\left(1-\frac{s-1}{n-k-1}\right)^{d-1}}, \tag{3.16}$$

where we used that $\binom{n-k}{s} = \frac{n-k}{s}\binom{n-k-1}{s-1}$ and

$$\frac{(k+d)\binom{n-k-d}{s-1}}{(n-k)\binom{n-k-1}{s-1}} \le \frac{k+d}{n-k}\left(\frac{n-k-s}{n-k-1}\right)^{d-1} = \frac{k+d}{n-k}\left(1-\frac{s-1}{n-k-1}\right)^{d-1} .$$

Let us denote $m = n-k-1$. Using (3.16), we get that for all $A \in \mathcal{X}\cup\mathcal{Y}$:

$$\begin{aligned}
&\sum_{S\subseteq[n]} X_S[\![A,A]\!] = \\
&\sum_{S\subseteq[n]\colon |S|=1} X_S[\![A,A]\!] + \sum_{s=2}^{n-k-d+1}\ \sum_{S\subseteq[n]\colon |S|=s} X_S[\![A,A]\!] \\
&\le \sqrt{\frac{k+d}{d}} + \frac{1}{m}\sum_{s=2}^{m+1}\sqrt{\frac{k+d}{(s-1)/m}\left(1-\frac{s-1}{m}\right)^{d-1}} \\
&\le \sqrt{\frac{k+d}{d}} + \int_0^1 \sqrt{\frac{k+d}{p}(1-p)^{d-1}}\,\mathrm{d}p \\
&= \sqrt{\frac{k+d}{d}} + \sqrt{k+d}\,\mathrm{B}\big(1/2,(d+1)/2\big) \\
&= O\left(\sqrt{\frac{k+d}{d}}\right),
\end{aligned}$$

where B stands for the beta function. Here, we substituted $p = (s-1)/m$, used monotonicity of the function $p^{-1}(1-p)^{d-1}$, and applied a well-known asymptotic for the beta function.

In the light of to Proposition 2.11, the condition (3.11) can be restated as follows:

OBSERVATION 3.9. *The feasible solution to the adversary bound (3.10) constructed in the proof of Theorem 3.6 has the following irrelevant variables in the sense of Proposition 2.11:*

- *If the input A is in $\mathcal{X}$ (i.e., $|A| = k$), an input variable $S \subseteq [n]$ is irrelevant if $S \cap A \neq \emptyset$.*
- *If the input A is in $\mathcal{Y}$ (i.e., $|A| = k+d$), an input variable $S \subseteq [n]$ is irrelevant if $|S \cap A| \neq 1$.*

4 Quantum algorithm for junta testing.

The aim of this section is to prove the following theorem:

THEOREM 4.1. *There exists a bounded-error quantum tester that distinguishes k-juntas from functions that are ε-far from any k-junta, with query complexity*

$$O\left(\sqrt{k/\varepsilon}\log k\right).$$

Suppose $f\colon \{0,1\}^n \to \{\pm 1\}$ depends on a set $J \subseteq [n]$ of K variables. Thus, the promise is that either f is a k-junta ($K \le k$), or f is ε-far from any k-junta (we will call such f a "non-junta" for simplicity). The goal of the tester is to distinguish these two cases.

At the lowest level of our algorithm, there is the following subroutine.

LEMMA 4.2. (INFLUENCE TESTER) *There exists an algorithm that, given a subset $V \subseteq [n]$, accepts with probability at least 0.9 if $\mathrm{Inf}_V(f) \geq \delta$ and rejects with certainty if $\mathrm{Inf}_V(f) = 0$. The algorithm uses $O(\sqrt{1/\delta})$ queries and $O(n/\sqrt{\delta})$ other elementary operations.*

Proof. We pick x, y randomly as described in Section 2 below (2.6), and check if $f(x) \neq f(y)$. By applying amplitude amplification for $O(1/\sqrt{\delta})$ rounds to amplify the basis states where $f(x) \neq f(y)$, we obtain the lemma.

The idea is to run the GGT algorithm of Theorem 3.6 with the Influence Tester of Lemma 4.2 as the input oracle. The complication is that we do not know what value of δ we should specify: the Fourier weight of a non-junta can be either concentrated on few (though more than k) variables with large influence, or scattered over many variables with tiny influence, and these cases call for different values of δ. We identify roughly $\log k$ different types of non-juntas, and design a separate tester for each of them. A junta will be accepted (with high probability) by all of these testers, whereas a non-junta will be rejected by at least one of them. The description is given in Algorithm 1, followed by the definition of the different types of non-juntas.

Algorithm 1 Quantum Junta Tester

1. Accept if all of the following $\lfloor \log(200k) \rfloor + 2$ testers accept, reject if at least one of them rejects:
   - Tester of the first kind with $\ell \in \{0, \ldots, \lfloor \log(200k) \rfloor\}$;
   - Tester of the second kind.

Subroutine 1.1 Tester of the first kind

1. Run the GGT algorithm of Theorem 3.6 with parameters k and $d = 2^\ell$ and the following oracle:
   - On input $S \subseteq [n]$, run Influence Tester on $V = S$ and $\delta = \varepsilon/(2^{\ell+3} \log(400k))$.
2. Accept if the GGT algorithm accepts, otherwise reject.

Subroutine 1.2 Tester of the second kind

1. Estimate the acceptance probability of the following subroutine with additive error .05:
   - Generate $V \subseteq [n]$ by adding each i to V with probability $1/k$ independently at random.
   - Run Influence Tester with this choice of V and $\delta = \varepsilon/(4k)$.
2. Accept if the estimated acceptance probability is ≤ 0.8, otherwise reject.

Let us describe these types of non-juntas. For notational convenience, assume the first K variables are the influential ones, ordered by influence (of course, the tester does not know this order):

$$\begin{aligned} \mathrm{Inf}_1(f) \geq \mathrm{Inf}_2(f) \geq \cdots \geq \mathrm{Inf}_K(f) > 0 \\ = \mathrm{Inf}_{K+1}(f) = \cdots = \mathrm{Inf}_n(f). \end{aligned} \tag{4.17}$$

Our tester does not know the number $K - k$ of "extra" variables if f is far from any k-junta. However, Lemma 2.14 implies that

$$\mathrm{Inf}_{\{k+1,\ldots,K\}}(f) \geq \varepsilon. \tag{4.18}$$

Our tests are tailored to the following cases:

1. $\sum_{j=k+1}^{200k} \mathrm{Inf}_j(f) \geq \varepsilon/2$. This case is additionally split into $\lfloor \log(200k) \rfloor + 1$ subcases:
$$\left| \left\{ j \in [n] \;\middle|\; \mathrm{Inf}_j(f) \geq \frac{\varepsilon}{2^{\ell+3} \log(400k)} \right\} \right| \geq k + 2^\ell,$$
where $\ell \in \{0, \ldots, \lfloor \log(200k) \rfloor\}$. Such an f is *of the first kind*, for this value of ℓ.
2. $\sum_{j=k+1}^{200k} \mathrm{Inf}_j(f) \leq \varepsilon/2$. Such an f is *of the second kind*.

Note that f may be a non-junta of the first kind for many different values of ℓ simultaneously; an extreme example is if f is the n-bit parity function.

LEMMA 4.3. *Every non-junta f satisfies at least one of the cases above.*

Proof. It is clear that any f satisfies the first or the second case above, so the only thing we need to show is that the first case is fully covered by its $\lfloor \log(200k) \rfloor + 1$ subcases. Assume f satisfies the first case. Denote $\varepsilon' = \varepsilon/(8 \log(400k))$ and consider the following intervals, which together partition the interval $[0, 1]$:

$$A_\infty = \left[0,\ \frac{\varepsilon'}{2^{\lfloor \log(200k) \rfloor}}\right), A_\ell = \left[\frac{\varepsilon'}{2^\ell},\ \frac{\varepsilon'}{2^{\ell-1}}\right), A_0 = [\varepsilon',\ 1],$$

where ℓ runs from $\lfloor \log(200k) \rfloor$ to 1. Let

$$B_\ell = \left\{ j \in \{k+1, \ldots, 200k\} \mid \mathrm{Inf}_j(f) \in A_\ell \right\}.$$

Each j is included in exactly one of the B_ℓ. Let also $W_\ell = \sum_{j \in B_\ell} \mathrm{Inf}_j(f)$. Thus, $\sum_\ell W_\ell \geq \varepsilon/2$, because we are in the first case. Next, $W_\infty < 200k \cdot \varepsilon/(8 \cdot 2^{\lfloor \log(200k) \rfloor}) \leq \varepsilon/4$. Thus, there exists $\ell \in \{0, \ldots, \lfloor \log(200k) \rfloor\}$ such that $W_\ell \geq \varepsilon/(4 \log(400k))$. Then, either $\ell = 0$ and $|B_\ell| \geq 1$, or

$$|B_\ell| \geq W_\ell \Big/ \left(\frac{\varepsilon}{2^{\ell+2} \log(400k)}\right) \geq 2^\ell .$$

Also, all $j \in B_\ell$ satisfy $\mathrm{Inf}_j(f) \geq \varepsilon/(2^{\ell+3} \log(400k))$. By (4.17), all $j \in [k]$ also satisfy this inequality. This means that f satisfies the first case with this value of ℓ.

LEMMA 4.4. *For each $\ell \in \{0, \ldots, \lfloor \log(200k) \rfloor\}$, Subroutine 1.1 accepts if f is a junta, and rejects if f is a non-junta of the first kind for this value of ℓ. Its query complexity can be made $O(\sqrt{(k/\varepsilon)\log k})$.*

Proof. The composition in Subroutine 1.1 is understood here in the sense of Definition 2.10 with the functions F and G defined as follows.

The partial function F is the EGGT function from Definition 3.1. Given a function $h\colon 2^{[n]} \to \{0,1\}$, $F(h) = 0$ if $h = \mathrm{Intersects}_A$ with $|A| = k$, and $F(h) = 1$ if $h = \mathrm{Intersects}_A$ with $|A| = k + d$, where $\mathrm{Intersects}_A$ is defined in (3.7). In all other cases, the value $F(h)$ is not defined.

For each $S \subseteq [n]$, the partial function G_S is as defined in Lemma 4.2. Given a total function $f\colon \{0,1\}^n \to \{0,1\}$, define $G_S(f) = 0$ if $\mathrm{Inf}_S(f) = 0$ and $G_S(f) = 1$ if $\mathrm{Inf}_S(f) \geq \delta$. If $0 < \mathrm{Inf}_S(f) < \delta$, the value $G_S(f)$ is not defined.

The function evaluated in Subroutine 1.1 is the following restriction of the composed function $F \circ (G_\emptyset, G_{\{1\}}, G_{\{2\}}, \ldots, G_{[n]})$:

$$f \mapsto F\big(G_\emptyset(f), G_{\{1\}}(f), G_{\{2\}}(f), \ldots, G_{[n]}(f)\big),$$

where, as the arguments of F, we have all possible 2^n functions G_S. The composition here is understood as in Definition 2.10 with the irrelevant variables of F given by Observation 3.9.

The query complexity of the subroutine can be computed using Corollary 2.12. The complexity of the algorithm for F in Theorem 3.6 is $O(\sqrt{k/2^\ell})$, as $2^\ell = O(k)$. The quantum query complexity of each G_V is $O(\sqrt{(2^\ell/\varepsilon)\log k})$ by Lemma 4.2. Thus, the total query complexity of the subroutine is $O(\sqrt{(k/\varepsilon)\log k})$.

Let us prove the correctness of the subroutine. Assume f is a non-junta of the first kind with this value of ℓ. By definition, there exists $A \subseteq [n]$ of size $k + d$ such that for all $j \in A$, $\mathrm{Inf}_j(f) \geq \delta$. As the influence is monotone in S, $\mathrm{Inf}_S(f) \geq \delta$ for all S that intersect A. By Observation 3.9, all input variables S satisfying $S \cap A = \emptyset$ are irrelevant, hence the value of the composed function is 1 in this case.

On the other hand, if f is a junta, there exists $A \subseteq [n]$ of size k such that for all $S \subseteq [n]$ satisfying $S \cap A = \emptyset$, we have $\mathrm{Inf}_S(f) = 0$. By Observation 3.9, all input variables S satisfying $S \cap A \neq \emptyset$ are irrelevant, hence the value of the composed function is 0 in this case.

From the proof of Lemma 4.4, it is clear why we need a separate tester for the second case. If 2^ℓ becomes $\omega(k)$, the complexity of Influence Tester still grows as $\widetilde{O}(2^\ell/\varepsilon)$, whereas the GGT algorithm cannot use fewer than $O(1)$ queries. Our second tester (Subroutine 1.2) does not use the GGT algorithm, and relies on more traditional means.

LEMMA 4.5. *Subroutine 1.2 accepts if f is a junta, and rejects if f is a non-junta of the second kind. Its query complexity is $O(\sqrt{k/\varepsilon})$.*

Proof. The estimate of the query complexity of Subroutine 1.2 is straightforward. Let us prove its correctness. We will show that the inner procedure has acceptance probability ≤ 0.75 if f is a k-junta, and acceptance probability ≥ 0.85 if f satisfies the second case.

If f is a k-junta then the probability that the set V does not intersect with the set J of (at most k) relevant variables is:

$$(1 - 1/k)^{|J|} \geq (1 - 1/k)^k \geq 1/4,$$

assuming $k \geq 2$. If V and J are disjoint, then the algorithm always rejects, hence, the acceptance probability is at most 0.75.

Now suppose f is a non-junta of the second kind. For notational convenience, we still assume that the variables of f are ordered by decreasing influence as in (4.17). For $j \in [n]$, let us define

$$\underline{\mathrm{Inf}}_j(f) = \begin{cases} 0, & \text{if } j \leq 200k; \\ \sum_{S\colon S \cap \{200k+1,\ldots,j\} = \{j\}} \widehat{f}(S)^2, & \text{otherwise.} \end{cases}$$

For $S \subseteq [n]$, define $\underline{\mathrm{Inf}}_S(f) = \sum_{j \in S} \underline{\mathrm{Inf}}_j(f)$.

This quantity satisfies two important properties. First, $0 \leq \underline{\mathrm{Inf}}_S(f) \leq \mathrm{Inf}_S(f)$ for all $S \subseteq [n]$. And second, it is additive is S, i.e., $\underline{\mathrm{Inf}}_{S \cup T}(f) = \underline{\mathrm{Inf}}_S(f) + \underline{\mathrm{Inf}}_T(f)$ for all disjoint S and T. Note that $\mathrm{Inf}_S(f)$ is only subadditive in S.

Next, as f satisfies the second case, $\mathrm{Inf}_j(f) \leq \varepsilon/(200k)$ for $j > 200k$. Hence, $\underline{\mathrm{Inf}}_j(f) \leq \varepsilon/(200k)$ for all $j \in [n]$. Finally,

$$\begin{aligned} \underline{\mathrm{Inf}}_{[n]}(f) &= \mathrm{Inf}_{\{200k+1,\ldots,K\}}(f) \\ &\geq \mathrm{Inf}_{\{k+1,\ldots,K\}}(f) - \sum_{j=k+1}^{200k} \mathrm{Inf}_j(f) \geq \frac{\varepsilon}{2}\,. \end{aligned}$$

Consider the random variable $\underline{\mathrm{Inf}}_V(f)$ where V is as in Subroutine 1.2. Its expectation is

$$\mu = \mathbb{E}[\underline{\mathrm{Inf}}_V(f)] = \frac{1}{k}\underline{\mathrm{Inf}}_{[n]}(f) \geq \frac{\varepsilon}{2k}\,,$$

and its variance is

$$\begin{aligned} \sigma^2 = \mathrm{Var}[\underline{\mathrm{Inf}}_V(f)] &\leq \frac{1}{k}\sum_j \underline{\mathrm{Inf}}_j(f)^2 \\ &\leq \frac{1}{k}\max_j \underline{\mathrm{Inf}}_j(f) \cdot \underline{\mathrm{Inf}}_{[n]}(f) \leq \frac{\varepsilon}{200k}\mu \leq \frac{\mu^2}{100}\,. \end{aligned}$$

Then, Chebyshev's inequality implies

$$\begin{aligned}\Pr\big[\underline{\mathrm{Inf}}_V(f) < \varepsilon/4k\big] &\le \Pr\big[|\underline{\mathrm{Inf}}_V(f) - \mu| \ge \mu/2\big] \\ &\le \Pr\big[|\underline{\mathrm{Inf}}_V(f) - \mu| \ge 5\sigma\big] \le= 0.04.\end{aligned}$$

Hence, with probability at least 0.96, we have $\mathrm{Inf}_V(f) \ge \underline{\mathrm{Inf}}_V(f) \ge \varepsilon/4k$. If this is indeed the case, the influence tester in Lemma 4.2 accepts with probability ≥ 0.9. Thus, the inner procedure accepts with probability at least $0.96 \cdot 0.9 > 0.85$ if f satisfies the second case.

From Lemmas 4.4 and 4.5, it is easy to see that Algorithm 1 is correct. If f is a junta, then all of the $O(\log k)$ subtesters accept. If f is a non-junta, then at least one of them rejects (and the output of the remaining ones is not defined). Thus, our algorithm is of the "robust conjunction" from (2.5). Hence, using Example 2.13 and Corollary 2.12, we get that the query complexity of Algorithm 1 is

$$O\big(\sqrt{\log k} \cdot \sqrt{(k/\varepsilon)\log k}\big) = O\big(\sqrt{k/\varepsilon}\log k\big).$$

This concludes the proof of Theorem 4.1.

5 Efficient implementation.

The main aim of this section is to prove that the algorithm from Theorem 3.6 can be implemented time-efficiently. Here by "time" we mean the total number of gates the algorithm uses, both the query-gates and all elementary quantum gates (from some arbitrary fixed universal set of gates) used to implement the unitaries in between the queries. Moreover, we will prove that our algorithm computes a function that has irrelevant variables as specified by Observation 3.9.

For clarity, we will now explicitly describe the problem which arises from applying Definition 2.9 to the EGGT problem of Definition 3.1.

Definition 5.1. (QGGT) *In the* quantum gap group testing (QGGT) problem *with parameters k and d, one is given access to an oracle O_f satisfying the following properties. The oracle O_f acts on two registers: the n-qubit input register I, and an arbitrary internal working register W. The oracle is in the block-diagonal form $O_f = \bigoplus_{S\subseteq\{0,1\}^n} O_{f,S}$, where $O_{f,S}$ is a unitary operator on W, that gets invoked in O_f when the value of the register I is S. We are promised that O_f belongs to one of the following two families:*

$$\begin{aligned}\widetilde{\mathcal{X}} = \Big\{ O_f \;\Big|\; &\exists A \in \mathcal{X}\ \forall S \subseteq [n]: \\ &S \cap A = \emptyset \implies O_{f,S}|0\rangle_{\mathsf{W}} = |0\rangle_{\mathsf{W}} \Big\}\end{aligned} \tag{5.19}$$

and[5]

$$\begin{aligned}\widetilde{\mathcal{Y}} = \Big\{ O_f \;\Big|\; &\exists B \in \mathcal{Y}\ \forall S \subseteq [n]: \\ &S \cap B \neq \emptyset \implies O_{f,S}|0\rangle_{\mathsf{W}} = -|0\rangle_{\mathsf{W}} \Big\}.\end{aligned} \tag{5.20}$$

The task is to detect whether $O_f \in \widetilde{\mathcal{X}}$ or $O_f \in \widetilde{\mathcal{Y}}$.

Theorem 5.2. *There exists a quantum algorithm that solves the QGGT problem with parameters k and d in time $\widetilde{O}\big(n\sqrt{1+k/d}\big)$ using $O\big(\sqrt{1+k/d}\big)$ queries.*

The time complexity of the algorithm is roughly n times its query complexity; as mentioned in the introduction, this is probably the best one can hope for.

Note that the QGGT problem incorporates the usual quantization of the GGT problem from Definition 3.2. However, the QGGT problem is more general than the GGT problem, the difference being that $O_{f,S}$ may be an *arbitrary* unitary in W when the premises in (5.19) or (5.20) do not hold.

With Theorem 5.2 in hand, it is easy to show that Algorithm 1 can be implemented time-efficiently as well, with a slight increase in the number of queries.

Theorem 5.3. *There exists a bounded-error quantum tester that distinguishes k-juntas from functions that are ε-far from any k-junta in time $\widetilde{O}\big(n\sqrt{k/\varepsilon}\big)$ using $\widetilde{O}\big(\sqrt{k/\varepsilon}\big)$ queries.*

Proof. It is easy to see from the proof of Lemma 4.5 that the time complexity of Subroutine 1.2 is $\widetilde{O}\big(n\sqrt{k/\varepsilon}\big)$. Unfortunately, it is hard to estimate the time complexity of the implementation of Subroutine 1.1 in Lemma 4.4, because Lemma 4.4 invokes Proposition 2.11(b) to analyze the composition of quantum algorithms (Proposition 2.11(b) upper bounds the *query* complexity of the composition but not its *time* complexity). However, Algorithm 1 can be implemented to have time complexity $\widetilde{O}\big(n\sqrt{k/\varepsilon}\big)$ as follows.

We first reduce the error probability of each call to the Influence Tester of Lemma 4.2 to $\ll \varepsilon/k$ by $O(\log \frac{k}{\varepsilon})$ repetitions, and run it backwards (after copying the answer) to set the workspace back to its initial state; then run the QGGT algorithm on this oracle as if it's errorless. Standard techniques show that the resulting variant of Subroutine 1.1 can be made to have error probability $\le 1/3$, and we do not need to invoke Proposition 2.11(b) anymore. The query complexity of Subroutine 1.1 has now gone up by a factor $O(\log \frac{k}{\varepsilon})$, but its

[5] One can also weaken the premise $S \cap B \neq \emptyset$ in (5.20) to $|S \cap B| = 1$. We chose this definition to make the QGGT problem more similar to the GGT problem.

time complexity becomes $\widetilde{O}\big(n\sqrt{k/\varepsilon}\big)$, because it is the time complexity of the QGGT algorithm, plus its oracle-query complexity multiplied by the time complexity of the amplified Influence Tester of Lemma 4.2 that implements one oracle call.

The resulting variant of Algorithm 1 can thus be implemented in time $\widetilde{O}\big(n\sqrt{k/\varepsilon}\big)$.

5.1 Proof of Theorem 5.2. The problem is solved by a (by now relatively standard) implementation of the dual adversary bound as in [42]. The analysis follows [38], with the simplification that we have Boolean input and output (see also [8, Section 3.4]). Our main innovation here is an efficient implementation of a specific reflection in Section 5.2, which we do by means of a new and efficient quantum Fourier transform.

Recall the QGGT problem as defined in Definition 5.1. Due to technical reasons, we have to assume that $O_{f,S}$ not only satisfies (5.19) or (5.20), but is also a reflection. This is without loss of generality.

Our algorithm only uses the input register I, so we omit this subscript below. The register W is not written, but assumed to be in the state $|0\rangle_{\mathsf{W}}$. We also add a new basis state $|0\rangle$ to I, and assume that $O_f|0\rangle = -|0\rangle$ for all O_f.

The query-efficient algorithm in Theorem 3.6 was obtained by constructing the matrices X_S in (3.12) that depend on parameters α_s and β_s satisfying (3.13). The objective value (3.10a) is $W = O(\sqrt{1+k/d})$ by Claim 3.8. Also we define $\gamma = C_1\sqrt{W}$ for some constant C_1 to be determined later.

Let Λ be the projector onto the span of the vectors

$$\psi_A = |0\rangle + \gamma \sum_{s=1}^{n-k-d+1} \alpha_s \sum_{S\subseteq[n]\colon |S|=s,\ S\cap A=\emptyset} |S\rangle \tag{5.21}$$

over all $A \in \mathcal{X}$, and $R_\Lambda = 2\Lambda - I$ be the corresponding reflection. (In Section 5.2 we show how to implement R_Λ efficiently.) The QGGT problem is solved by Algorithm 2, where C is some constant to be defined later.

Algorithm 2 Quantum Algorithm for the QGGT problem

1. Prepare the state $|0\rangle$.
2. Perform phase estimation on the operator $U = O_f R_\Lambda$ with precision $\delta = 1/(CW)$.
3. Accept if and only if the phase-estimate is greater than δ.

Claim 5.4. *Algorithm 2 is correct.*

Proof. Let us first assume that $O_f \in \widetilde{\mathcal{Y}}$. Let $B \in \mathcal{Y}$ be a corresponding element from (5.20), so $|B| = k+d$. Define the following vector

$$u = \gamma|0\rangle - \sum_{s=1}^{n-k-d+1} \beta_s \sum_{S\subseteq[n]\colon |S|=s,\ |S\cap B|=1} |S\rangle. \tag{5.22}$$

The squared norm of this vector is

$$\begin{aligned}\|u\|^2 &= \gamma^2 + \sum_{s=1}^{n-k-d+1} (k+d)\binom{n-k-d}{s-1}\beta_s^2 \\ &= \gamma^2 + \sum_{S\subseteq[n]} X_S[\![B,B]\!] \le C_1^2 W + W,\end{aligned}$$

where the second equality uses (3.15), and the last inequality uses that the objective value (3.10a) is W.

We now show that u is an eigenvector of $U = O_f R_\Lambda$ with eigenvalue 1 (so the eigenvalue's phase is 0). First, $O_f u = -u$, because $O_f|S\rangle = -|S\rangle$ for all S occurring in (5.22) and we earlier already assumed $O_f|0\rangle = -|0\rangle$. Second, for all $A \in \mathcal{X}$ we have

$$\begin{aligned}\langle \psi_A, u\rangle &= \gamma - \gamma \sum_{s=1}^{n-k-d+1} \sum_{S\subseteq[n]:|S|=s,S\cap A=\emptyset,|S\cap B|=1} \alpha_s\beta_s \\ &= \gamma - \gamma \sum_{S\colon A\cap S=\emptyset \text{ xor } B\cap S=\emptyset} X_S[\![A,B]\!] = 0,\end{aligned}$$

where we used (3.12) and (3.10b). Hence, $\Lambda u = 0$ and $R_\Lambda u = (2\Lambda - I)u = -u$. Therefore, $Uu = O_f R_\Lambda u = u$.

Furthermore, the inner product of the normalized eigenvector $u/\|u\|$ and $|0\rangle$ is

$$\frac{\gamma}{\|u\|} \ge \frac{C_1\sqrt{W}}{\sqrt{C_1^2 W + W}} = \frac{1}{\sqrt{1+1/C_1^2}},$$

which can be made arbitrarily close to 1 by setting C_1 to a sufficiently large constant. Since $|0\rangle$ is the starting state of Algorithm 2, the algorithm will (with probability at least 2/3 if we set C_1 appropriately) produce a phase estimate that is at most δ, and correctly rejects $O_f \in \widetilde{\mathcal{Y}}$.

Now assume $O_f \in \widetilde{\mathcal{X}}$. Let $A \in \mathcal{X}$ be the corresponding element from (5.19), so $|A| = k$. In this case, we will apply Lemma 2.3 with $R_1 = -R_\Lambda = I - 2\Lambda$, $R_2 = -O_f$ (hence $U = O_f R_\Lambda = R_2 R_1$), $\Pi_1 = I - \Lambda$, $\Pi_2 = (I - O_f)/2$, and $w = \psi_A$. Indeed, since $\Lambda w = w$, w lies in the kernel of Π_1, and we assume O_f is a reflection, so the conditions of the lemma are satisfied. We have $\Pi_2 w = |0\rangle$ because $O_f|0\rangle = -|0\rangle$, and $O_f|S\rangle = |S\rangle$ for all S in the support of ψ_A. Also,

$$\begin{aligned}\|\psi_A\|^2 &= 1 + \gamma^2 \sum_{s=1}^{n-k-d+1} \binom{n-k}{s}\alpha_s^2 \\ &= 1 + \gamma^2 W = 1 + C_1^2 W^2.\end{aligned}$$

Since also $W = \Omega(1)$, we have $\|w\| = O(W)$. Therefore, using Lemma 2.3, the algorithm's initial state $|0\rangle$ barely overlaps with eigenvectors of $U = R_2R_1$ whose phase is (2δ)-close to 0:

$$\|P_{2\delta}|0\rangle\| = \|P_{2\delta}\Pi_2 w\| \le \delta\|w\| = O(1/C).$$

Hence the probability that phase estimation erroneously yields an estimate that is δ-close to 0 can be made less than 1/3 by choosing C a sufficiently large constant. Then Algorithm 2 accepts all $O_f \in \widetilde{\mathcal{X}}$ with probability at least 2/3.

As R_Λ can be implemented without executing the input oracle, the query complexity of Algorithm 2 is $O(W) = O(\sqrt{1+k/d})$ by Theorem 2.2. To get the time complexity, the query complexity has to be multiplied by the cost of implementing $U = O_f R_\Lambda$. In Section 5.2 we show that R_Λ can be implemented in time $\widetilde{O}(n)$. Thus, Algorithm 2 can be implemented in time $\widetilde{O}(n\sqrt{1+k/d})$.

5.2 Efficient implementation of R_Λ. This section is devoted to the proof of the following lemma, which shows that the reflection $R_\Lambda = 2\Lambda - I$ can be implemented efficiently, in time $\widetilde{O}(n)$. For simplicity we assume $n > 2k$. This is without loss of generality, as we can extend the set $[n]$ with dummy elements. Next, we identify $|0\rangle$ of Eq. (5.21) with $|\emptyset\rangle$, and absorb γ into α_s. To state the lemma, it is also more convenient to replace A in Eq. (5.21) by its complement, $T = [n] \setminus A$.

LEMMA 5.5. *Let $\alpha_0, \alpha_1, \ldots, \alpha_{n-k}$ be arbitrary complex numbers and let Λ be the projector onto the span of the vectors*

$$\psi_T = \sum_{\ell=0}^{n-k} \alpha_\ell \sum_{B\subseteq T:\ |B|=\ell} |B\rangle$$

over all $T \subseteq [n]$ with $|T| = n-k$. Then, the corresponding reflection $R_\Lambda = 2\Lambda - I$ can be implemented in time $\widetilde{O}(n)$, up to an error in the operator norm that can be made smaller than any inverse polynomial in n.

Representation theory background. In order to prove Lemma 5.5, we will use the structural properties implied by the invariance of the vectors ψ_T under permutations of $[n]$. We need some basic results from the representation theory of the symmetric group. These results are only used in this section. The reader may refer to a textbook on the topic such as [44], or to the appendix of [9], where we briefly formulate the required notions and results.

Let $\mathbb{S}_n$ denote the symmetric group on $[n]$. We consider (left) modules over the group algebra $\mathbb{C}\mathbb{S}_n$. We call them $\mathbb{S}_n$-*modules*; they are also known as *representations* of $\mathbb{S}_n$. There is a 1-1 correspondence between irreducible $\mathbb{S}_n$-modules and partitions $(t_1, \ldots, t_k)$ of n (where $t_1 \ge t_2 \ge \cdots \ge t_k$ and $t_1 + t_2 + \cdots + t_k = n$). Irreducible $\mathbb{S}_n$-modules are called *Specht modules*.

A linear mapping $\theta\colon V \to W$ between two $\mathbb{S}_n$-modules is called an $\mathbb{S}_n$-*homomorphism* if, for all $\pi \in \mathbb{S}_n$ and $v \in V$, we have $\theta(\pi v) = \pi(\theta(v))$. The following result is basic for such homomorphisms:

LEMMA 5.6. (SCHUR'S LEMMA) *Assume $\theta\colon V \to W$ is an $\mathbb{S}_n$-homomorphism between two irreducible $\mathbb{S}_n$-modules V and W. Then, $\theta = 0$ if V and W are not isomorphic. Otherwise, θ is uniquely defined up to a scalar multiplier.*

Let M denote the complex vector space with the set of subsets of $[n]$ as its orthonormal basis and with the group action $\pi A = \pi(A)$, where $\pi \in \mathbb{S}_n$, $A \subseteq [n]$, and $\pi(A)$ denotes the image of the set A under the transformation π. We call $\{A\}_{A\subseteq[n]}$ the *standard basis* of M.

The module M naturally decomposes into a direct sum $M = \bigoplus_{\ell=0}^{n} M_\ell$, where M_ℓ is spanned by the subsets of cardinality ℓ.[6] The following lemma describes the decomposition of M_ℓ into irreducible submodules $S_\ell(t)$ (for different values of t), which will be isomorphic to the Specht module $S(t)$ corresponding to the partition $(n-t, t)$ of n.

In the formulation of the lemma and later we use $\otimes$ to denote disjoint union of subsets of $[n]$, extended by linearity, so for example $(\{1\} - \{2\}) \otimes (\{3\} - \{4\}) = \{1,3\} - \{1,4\} - \{2,3\} + \{2,4\}$.

LEMMA 5.7. *The $\mathbb{S}_n$-module M_ℓ has the following decomposition into irreducible submodules: $M_\ell = \bigoplus_{t=0}^{\ell'} S_\ell(t)$, where $\ell' = \min\{\ell, n-\ell\}$, and each $S_\ell(t)$ is isomorphic to $S(t)$. The submodule $S_\ell(t)$ is spanned by the vectors*

(5.23)

$$v_\ell(t,a,b) = \big(\{a_1\} - \{b_1\}\big) \otimes \cdots \otimes \big(\{a_t\} - \{b_t\}\big) \otimes \Bigg(\sum_{A\subseteq[n]\setminus\{a_1,\ldots,a_t,b_1,\ldots,b_t\}:\ |A|=\ell-t} A \Bigg)$$

defined by disjoint sequences $a = (a_1, \ldots, a_t)$ and $b = (b_1, \ldots, b_t)$ of pairwise distinct elements of $[n]$. The dimension of $S(t)$ is $\binom{n}{t} - \binom{n}{t-1}$.

There is a unique (up to a scalar) $\mathbb{S}_n$-isomorphism between $S_\ell(t)$ and $S_m(t)$. We can choose the scalar so that the isomorphism maps each vector $v_\ell(t,a,b)$ to the corresponding $v_m(t,a,b)$.

[6] In terms of [44], M_ℓ is isomorphic to the permutation module corresponding to the partition $(n-\ell', \ell')$ of n, where $\ell' = \min\{\ell, n-\ell\}$.

The lemma follows from general theory [44, Sections 2.9 and 2.10]. The Appendix of [9] contains a short proof, see also Remark 5.11 below. Figure 1 depicts the different subspaces involved in the decomposition of M.

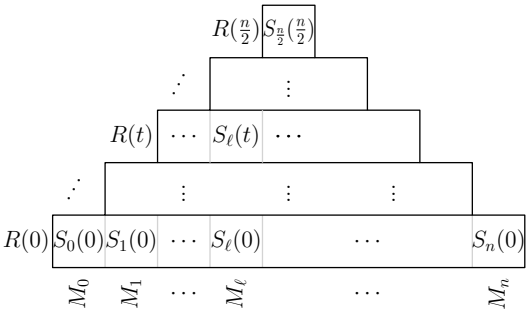

Figure 1: Decomposition of M

Let $\tilde{v}_\ell(t,a,b)$ denote the normalized vector $v_\ell(t,a,b)/\|v_\ell(t,a,b)\|$, that is

$$v_\ell(t,a,b) = \sqrt{2^t \binom{n-2t}{\ell-t}}\, \tilde{v}_\ell(t,a,b). \tag{5.24}$$

Also, let $\vartheta_{t\to\ell}\colon S_t(t) \to S_\ell(t)$ denote the isomorphism from Lemma 5.7 given by $\tilde{v}_t(t,a,b) \mapsto \tilde{v}_\ell(t,a,b)$. This is a unitary transformation.

For each t, we choose an orthonormal basis $\{e_t(t,x)\}_x$ of $S_t(t)$. Also, let $e_\ell(t,x) = \vartheta_{t\to\ell}\ e_t(t,x)$, so that $\{e_\ell(t,x)\}_x$ is an orthonormal basis of $S_\ell(t)$. The precise choice of the basis of $S_t(t)$ is irrelevant, but it is important that the bases of $S_\ell(t)$ for various ℓ are synchronized via the isomorphism $\vartheta_{t\to\ell}$. The set $\{e_\ell(t,x)\}_{\ell,t,x}$ forms an orthonormal basis of M, which we call the *Fourier basis*. Let $R(t)$ denote the submodule $\bigoplus_{\ell=t}^{n-t} S_\ell(t)$ of M.

Claim 5.8. *In the Fourier basis, any $\mathbb{S}_n$-homomorphism from M to itself is of the form $\bigoplus_{t=0}^{\lfloor n/2\rfloor} A_t \otimes I_{S(t)}$, where $A_t \otimes I_{S(t)}$ acts on $R(t)$, A_t is an $(n-2t+1)\times(n-2t+1)$ matrix, and $I_{S(t)}$ denotes the identity operator on $S(t)$.*

Proof. By Schur's lemma, any $\mathbb{S}_n$-homomorphism maps each irreducible module to an isomorphic one. Hence, each $R(t)$ is mapped to itself. Also, as $\vartheta_{t\to\ell}$ is the only isomorphism between $S_t(t)$ and $S_\ell(t)$, we see by Schur's lemma that a vector $e_\ell(t,x)$ is mapped into a linear combination of the vectors $\{e_m(t,x)\}_m$. Thus, in $R(t)$, the homomorphism has the form $A_t \otimes I_{S(t)}$ for some A_t.

The quantum Fourier transform of M. Let the register A have M as its vector space with the standard basis. Let also T and L be $(n+1)$-qudits (i.e., registers of dimension $n+1$), and let n-qubit register B store indices x of the Fourier basis elements $e_\ell(t,x)$. The *Fourier transform* of M is the following unitary map, for which it is convenient to use ket notation:

$$F\colon |t\rangle_\mathsf{T}|\ell\rangle_\mathsf{L}|x\rangle_\mathsf{B} \mapsto |e_\ell(t,x)\rangle_\mathsf{A}. \tag{5.25}$$

Note that while F is a unitary from a 2^n-dimensional space to a 2^n-dimensional space, it is convenient to use more than n qubits to represent the basis states (namely $n + 2\lceil\log(n+1)\rceil$ qubits).

The Fourier transform segregates copies of non-isomorphic Specht modules of M by assigning them different values of t in the register T. For each Specht module $S(t)$, the copies are labeled by ℓ, which is the size (as a subset of $[n]$) of the standard basis elements of M used by the copy. Finally, the basis elements of $S(t)$ are indexed by x, the precise choice of which is irrelevant for our application. The next theorem follows from the efficient quantum Schur-Weyl transform of Bacon, Chuang and Harrow [5, 6]. See the full version of our paper for a self-contained proof.

Theorem 5.9. *There exists a quantum algorithm with time complexity $\widetilde{O}(n)$ that implements the map from (5.25) for all choices of t, ℓ and x for which the last expression is defined, up to an error in the operator norm that can be made smaller than any inverse polynomial in n.*

Decomposing Λ in terms of representations. The main observation behind our implementation of R_Λ is that Λ is invariant under the permutation of elements, hence it is an $\mathbb{S}_n$-homomorphism from M to itself, and thus subject to the decomposition of Claim 5.8. In fact, it is not hard to obtain the matrices A_t in this decomposition.

Claim 5.10. *The image of Λ contains at most one copy of each $S(t)$ for $t = 0,\dots,k$. In the Fourier basis, Λ has the following form: $\bigoplus_{t=0}^{k}(\tilde{w}_t\tilde{w}_t^*)\otimes I_{S(t)}$, where $\tilde{w}_t$ is the normalized version of the $(n-2t+1)$-dimensional vector w_t that is given by*

$$w_t[\![\ell]\!] = \alpha_\ell \binom{n-\ell-t}{k-t} \sqrt{\binom{n-2t}{\ell-t}}, \tag{5.26}$$

for $\ell \in \{t,\dots,n-k\}$, and $w_t[\![\ell]\!] = 0$ for $\ell \in \{n-k+1,\dots,n-t\}$. (If w_t is the 0-vector, we assume that $\tilde{w}_t$ is the 0-vector as well.)

Proof. We first assume that $\alpha_{n-k} \neq 0$. Because Λ is an $\mathbb{S}_n$-homomorphism, by Claim 5.8, Λ has the form $\bigoplus_t A_t \otimes I_{S(t)}$ in the Fourier basis. We claim

that Λ contains exactly one copy of each $S(t)$ with $0 \le t \le k$, i.e., all A_t are rank-1 projectors: indeed, the projection of Λ on M_{n-k} is surjective, and as we know from Lemma 5.7, the latter does contain a copy of each $S(t)$. Since $S(t)$ has dimension $\binom{n}{t} - \binom{n}{t-1}$, the direct sum of these copies of $S(t)$ already has dimension $\sum_{t=0}^{k} \left(\binom{n}{t} - \binom{n}{t-1}\right) = \binom{n}{k}$. On the other hand, Λ clearly has dimension at most $\binom{n}{k}$, hence Λ cannot contain more than one copy of any of these $S(t)$. Thus, Λ actually has the form $\bigoplus_{t=0}^{k} (\tilde{w}_t \tilde{w}_t^*) \otimes I_{S(t)}$ for some vectors $\tilde{w}_t$.

It remains to find the coefficients of $\tilde{w}_t$. Take a vector $v_{n-k}(t,a,b) \in M_{n-k}$ for some sequences a and b, and act on it with the linear transformation that maps a basis vector $T \in M_{n-k}$ to the vector $\psi_T \in M$. The resulting vector u is clearly in the image of Λ. We claim that

$$\begin{aligned} u &= \sum_{\ell=t}^{n-k} \alpha_\ell \binom{n-\ell-t}{k-t} v_\ell(t,a,b) \\ &= \sum_{\ell=t}^{n-k} \alpha_\ell \binom{n-\ell-t}{k-t} \sqrt{2^t \binom{n-2t}{\ell-t}}\, \tilde{v}_\ell(t,a,b). \end{aligned} \tag{5.27}$$

To prove the first equality of (5.27), consider a basis element $A \in M_\ell$ for an $\ell \in \{t, \ldots, n-k\}$ and look at its coefficient in u. If A contains both a_i and b_i for some i, it appears in none of the ψ_T of which u consists. If A avoids both a_i and b_i for some i, then any coefficient it gets from ψ_T is cancelled by the coefficient it gets from $\psi_{T \triangle \{a_i, b_i\}}$, where $\triangle$ stands for symmetric difference. Finally, if A uses exactly one of each a_i, b_i, it appears in ψ_T for $\binom{n-\ell-t}{k-t}$ choices of T, with coefficient equal to its coefficient in $v_\ell(t,a,b)$ times α_ℓ in each. This establishes the first equality of (5.27). The second equality in (5.27) follows immediately from (5.24).

Now let $v = (v_x)$ be the vector of coefficients of the representation of $\tilde{v}_t(t,a,b)$ in the basis $\{e_t(t,x)\}_x$, i.e., $\tilde{v}_t(t,a,b) = \sum_x v_x e_t(t,x) = F\big(|t\rangle_{\mathsf{T}} |t\rangle_{\mathsf{L}} |v\rangle_{\mathsf{B}}\big)$. Then, by our choice of orthonormal basis, $F^{-1}(\tilde{v}_\ell(t,a,b)) = |t\rangle_{\mathsf{T}} |\ell\rangle_{\mathsf{L}} |v\rangle_{\mathsf{B}}$ for $\ell \in \{t, \ldots, n-k\}$, and from (5.27), we have

$$\begin{aligned} &F^{-1}\left(u/\sqrt{2^t}\right) \\ &\quad = \sum_{\ell=t}^{n-k} \alpha_\ell \binom{n-\ell-t}{k-t} \sqrt{\binom{n-2t}{\ell-t}} F^{-1}(\tilde{v}_\ell(t,a,b)) \\ &\quad = |t\rangle_{\mathsf{T}} |w_t\rangle_{\mathsf{L}} |v\rangle_{\mathsf{B}}, \end{aligned}$$

where w_t is defined in (5.26). As u is in the image of Λ and $F^{-1}u$ is u represented in the Fourier basis, w_t must be proportional to $\tilde{w}_t$.

Now consider the case $\alpha_{n-k} = 0$. In this case, change α_{n-k} to an arbitrary non-zero value, and perform the above calculations for the resulting projector Λ'. The result follows from the observation that the image of Λ is a projection of the image of Λ' onto $\bigoplus_{\ell<n-k} M_\ell$.

REMARK 5.11. *Note that (5.27) essentially proves the second part of Lemma 5.7. Indeed, it shows that the transformation $v_{n-k}(t,a,b) \mapsto v_\ell(t,a,b)$ is linear (which is not obvious from (5.23)). It is clear that it is invariant under the action of $\mathbb{S}_n$, hence, it is an isomorphism between the copies of $S(t)$ in M_{n-k} and M_ℓ.*

Implementing R_Λ. Having the efficient implementation of the QFT of Theorem 5.9, it is easy to implement the translation R_Λ up to polynomially small error. First, we run the QFT of Theorem 5.9 in reverse, and obtain the representation of M in the Fourier basis. In this basis, by Claim 5.10, our task boils down to the reflection about the vector $\tilde{w}_t$ in the register L, where t is the value of the register T. The implementation of this reflection is reasonably straightforward. See the full version of the paper for detail.

6 Quantum lower bound for junta testing.

Let us assume that $\varepsilon = \Omega(1)$. Tight classical lower bounds on junta testing [25, 17] are based on distinguishing a k-junta from a function that depends on $k + O(1)$ variables. As noted by Atıcı and Servedio [4], this approach is doomed in the quantum setting because these two cases can be distinguished in $O(\log k)$ quantum queries as follows. For a function that depends on only $k + O(1)$ variables but is far from any k-junta, it follows from Lemma 2.14 that at least one of the $O(1)$ "extra" variables has $\Omega(1)$ influence. Hence there exists a subset $S \subseteq [n]$ of $k+1$ variables each having influence $\Omega(1)$. Each of those $k+1$ variables will occur in a Fourier Sample with constant probability, so the probability that a fixed variable from S is not seen in t Fourier Samples is exponentially small in t. By the union bound, after $t = O(\log k)$ Fourier Samples, with high probability all $k+1$ variables of S will have been seen and we can conclude the function is not a k-junta.

Instead of this, Atıcı and Servedio presented a different approach based on distinguishing a k-junta from a function that depends on $k + \Omega(k)$ variables. Using this technique, they proved an $\Omega(\sqrt{k})$ lower bound for a special class of *non-adaptive* quantum algorithms.

In this section, we give an explicit description of the Atıcı-Servedio construction, and use it to prove a

quantum lower bound for the junta testing problem. Consider the following problem.

DEFINITION 6.1. (TESTING THE IMAGE SIZE) *An* image size tester*, given oracle access to a function $g\colon [m] \to [n]$, is required to distinguish whether the image of g is of size at most ℓ, or g is ε-far away from any such function.*

It turns out that a junta tester can be used to solve this problem. The connection is through the following ancillary function.

DEFINITION 6.2. (ADDRESSING FUNCTION) *Assume that m is a power of two, and $g\colon [m] \to [n]$ is a function. We define the corresponding* addressing function *$f\colon \{0,1\}^{n+\log m} \to \{\pm 1\}$ as follows. Interpret the input string x of f as a concatenation yz with $y \in \{0,1\}^n$ and $z \in \{0,1\}^{\log m} = [m]$. Then, $f(x) = (-1)^{y_{g(z)}}$. The variables in y are called* addressed variables*, and the variables in z are called the* address variables*.*

It is easy to see that a quantum query to f can be simulated by two quantum queries to g: one to compute $g(z)$, and one to uncompute it.

LEMMA 6.3. *For a function $g\colon [m] \to [n]$ with m a power of 2, let $f\colon \{0,1\}^{n+\log m} \to \{\pm 1\}$ be the corresponding addressing function. Let $\ell \geq 1$ be an integer and define $k = \ell + \log m$. If the size of the image of g does not exceed ℓ, then f is a k-junta. Conversely, if g is ε-far from any function with an image of size at most ℓ, then f is $\varepsilon'/2$-far from any k-junta where $\varepsilon' = \varepsilon - (\log m)/k$.*

Proof. The first statement is obvious. So assume g is ε-far from any function with an image of size at most ℓ. We claim that g is also ε'-far from any function with an image of size at most k. Indeed, if h is a function with image of size at most k, we can reduce its image to be of size at most ℓ by modifying it on at most a $(\log m)/k$ fraction of inputs corresponding to the "least popular" outputs.

In order to show that f is $\varepsilon'/2$-far from any k-junta, take an arbitrary k-subset $W \subseteq [n + \log m]$, and any Boolean function h depending only on the variables in W. We want to show that f is $\varepsilon'/2$-far from h. Indeed, by the previous claim, at least ε' fraction of the inputs to g map to indices outside $W \cap [n]$. For any such $z \in [m]$, and any $y \in \{0,1\}^n$, exactly one $x \in \{yz, y^{\oplus g(z)}z\}$ satisfies $f(x) \neq h(x)$ (where $x^{\oplus j}$ stands for x with the jth bit flipped). Hence, the distance between f and h is at least $\varepsilon'/2$.

Let us now state some corollaries of this result. First, we get an upper bound on the quantum query complexity of testing the support size.

COROLLARY 6.4. *If $\log m = o(\ell)$, the image size can be tested in $O\big(\sqrt{\ell/\varepsilon}\log \ell\big)$ quantum queries.*

More importantly, however, we get a lower bound on the quantum query complexity of junta testing. This is based on the following well-known special case of the image size testing problem.

DEFINITION 6.5. (COLLISION PROBLEM [20]) *Let m be an even integer, and $n \in \mathbb{N}$. In the* collision problem*, one is given oracle access to a function $g\colon [m] \to [n]$, that is either 1-to-1 or 2-to-1. The task is decide which is the case.*

Brassard *et al.* [20] constructed a quantum $O(m^{1/3})$-query algorithm for the collision problem. Later, Aaronson and Shi [1] proved a matching lower bound:

THEOREM 6.6. *The bounded-error quantum query complexity of the collision problem is $\Omega(m^{1/3})$.*

If $g\colon [m] \to [n]$ is 2-to-1, then its image size is $m/2$ and the corresponding addressing function f depends on only $m/2 + \log m$ variables. On the other hand, if g is 1-to-1, then its image size is m and f depends on $m + \log m$ variables. Moreover, it follows from Lemma 6.3 that f is 1/5-far from any $(m/2 + \log m)$-junta if m is large enough. Combined with Theorem 6.6, we get

THEOREM 6.7. *Every quantum tester that distinguishes k-juntas from functions that are 1/5-far from any k-junta with bounded error, needs to make $\Omega(k^{1/3})$ queries to the function.*

7 Conclusion and open problems.

In this paper we presented quantum algorithms for testing several well known properties of Boolean functions. Our main result is a quantum algorithm for the k-junta testing problem with query complexity $O(\sqrt{k/\varepsilon}\log k)$, and a time-efficient implementation of this based on a new near-linear time implementation of a shallow version of the quantum Fourier transform over the symmetric group. The query complexity of our tester is almost quadratically better than the best previous quantum tester and also almost quadratically better than the best-possible classical tester.

The topics for future work include:

1. **Better lower bound for junta testing.** The main open question is: what is the actual quantum query complexity of this problem?

   We believe that the true answer is around $\sqrt{k/\varepsilon}$ but it is quite challenging to improve our current lower bound of $\Omega(k^{1/3})$. Nevertheless, we think that Lemma 6.3 may give a lower bound of $\Omega(k^{1/2-\delta})$ for

any $\delta > 0$. In particular, we think that it should be possible to combine the lower bound construction by Raskhodnikova *et al.* [40] with two recent developments in quantum lower bounds: Zhandry's new machinery for the polynomial method [47], which he applied to the collision and the set equality problems [48], and Belovs's and Rosmanis's tight adversary lower bounds for the same functions [13].

2. **Better upper bound.** Regarding the upper bound, we wonder if the $\log k$ factor can be removed. This question is essentially equivalent to finding a solution to the adversary bound for the GGT problem that works *for all values of d simultaneously*. By this, we mean a feasible solution to (3.10) such that, when f is an Intersects_A function,

$$\sum_{S \subseteq [n]} X_S[\![f, f]\!] = \begin{cases} O(\sqrt{k}), & \text{if } |A| = k; \\ O\left(\frac{\sqrt{k}}{d}\right), & \text{if } |A| = k + d > k. \end{cases}$$

   Note that our current solution does not satisfy this property because we use different rescaling for each value of d. A different approach may be needed to obtain this property.

3. **Other applications of QGGT and our QFT.** Several of our algorithms are based on a quantum algorithm for a group testing problem, QGGT, which we find quite interesting in its own right, as it shows a quartic quantum-over-classical speedup. We think there might be more applications for QGGT waiting to be found.

Acknowledgements. We thank Ashley Montanaro for getting some of us interested in quantum junta testers in the first place, and for initial discussions; Eric Blais for some discussions about monotonicity testing, and in particular for his suggestion to study the classical complexity of the GGT problem; Mark Zhandry for answering a question about [48]; Alexander Russell for helpful discussions about the quantum Fourier transform over the symmetric group; Aram Harrow for answering a question about the Schur-Weyl transform; and Jeroen Zuiddam for some helpful comments. We thank the anonymous referees for many helpful suggestions and bringing references [5, 31] to our attention.

References

[1] S. Aaronson and Y. Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM*, 51(4):595–605, 2004.

[2] A. Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64(4):750–767, 2002.

[3] A. Ambainis, K. Balodis, A. Belovs, T. Lee, M. Santha, and J. Smotrovs. Separations in query complexity based on pointer functions. 2015.

[4] A. Atıcı and R. A. Servedio. Quantum algorithms for learning and testing juntas. *Quantum Information Processing*, 6(5):323–348, 2007.

[5] D. Bacon, I. Chuang, and A. Harrow. Efficient quantum circuits for Schur and Clebsch-Gordan transforms. *Physical Review Letters*, 97(17):170502, 2006.

[6] D. Bacon, I. Chuang, and A. Harrow. The quantum Schur and Clebsch-Gordan transforms: I. efficient qudit circuits. In *Proc. of 18th ACM-SIAM SODA*, pages 1235–1244, 2007.

[7] A. Belovs. Span programs for functions with constant-sized 1-certificates. In *Proc. of 44th ACM STOC*, pages 77–84, 2012.

[8] A. Belovs. *Applications of the Adversary Method in Quantum Query Algorithms*. PhD thesis, University of Latvia, 2013.

[9] A. Belovs. Quantum algorithms for learning symmetric juntas via the adversary bound. *Computational Complexity*, 24(2):255–293, 2015.

[10] A. Belovs. Variations on quantum adversaries. 2015.

[11] A. Belovs and E. Blais. Quantum algorithm for monotonicity testing on the hypercube. 2015.

[12] A. Belovs and B. W. Reichardt. Span programs and quantum algorithms for st-connectivity and claw detection. In *Proc. of 20th ESA*, volume 7501 of *LNCS*, pages 193–204, 2012.

[13] A. Belovs and A. Rosmanis. Adversary lower bounds for the collision and the set equality problems. 2013.

[14] S. Ben-David. A super-Grover separation between randomized and quantum query complexities. 2015.

[15] E. Blais. Testing juntas nearly optimally. In *Proc. of 41st ACM STOC*, pages 151–158, 2009.

[16] E. Blais. Testing juntas: a brief survey. In Goldreich [32], pages 32–40.

[17] E. Blais, J. Brody, and K. Matulef. Property testing lower bounds via communication complexity. *Computational Complexity*, 21(2):311–358, 2012.

[18] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4-5):493–505, 1998.

[19] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *AMS Contemporary Mathematics Series*, pages 53–74, 2002.

[20] G. Brassard, P. Høyer, and A. Tapp. Quantum cryptanalysis of hash and claw-free functions. In *Proc. of 3rd LATIN*, volume 1380 of *LNCS*, pages 163–169. Springer, 1998.

[21] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288:21–43, 2002.

[22] A. Chakrabarti and O. Regev. An optimal lower bound on the communication complexity of gap Hamming distance. *SIAM Journal on Computing*, 41(5):1299–1317, 2012.

[23] Y. Cheng. An efficient randomized group testing procedure to determine the number of defectives. *Operations Research Letters*, 39(5):352–354, 2011.

[24] Y. Cheng and Y. Xu. An efficient FPRAS type group testing procedure to approximate the number of defectives. *Journal of Combinatorial Optimization*, 27(2):302–314, 2014.

[25] H. Chockler and D. Gutfreund. A lower bound for testing juntas. *Information Processing Letters*, 90(6):301–305, 2004.

[26] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354, 1998.

[27] P. Damaschke and A. S. Muhammad. Bounds for nonadaptive group tests to estimate the amount of defectives. In *Proc. of 4th COCOA*, volume 6509 of *LNCS*, pages 117–130. Springer, 2010.

[28] R. Dorfman. The detection of defective members of large populations. *The Annals of Mathematical Statistics*, 14(4):436–440, 1943.

[29] D. Z. Du and F. Hwang. *Combinatorial group testing and its applications*, volume 3 of *Series on Applied Mathematics*. World Scientific, 1993.

[30] E. Fischer, G. Kindler, D. Ron, S. Safra, and A. Samorodnitsky. Testing juntas. *Journal of Computer and System Sciences*, 68(4):753–787, 2004.

[31] D. García-Soriano. *Query-Efficient Computation in Property Testing and Learning Theory*. PhD thesis, CWI and University of Amsterdam, 2012.

[32] O. Goldreich, editor. *Property Testing: Current Research and Surveys*, volume 6390 of *LNCS*. Springer, 2010.

[33] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. of 28th ACM STOC*, pages 212–219, 1996.

[34] P. Høyer, T. Lee, and R. Špalek. Negative weights make adversaries stronger. In *Proc. of 39th ACM STOC*, pages 526–535, 2007.

[35] K. Iwama, H. Nishimura, R. Raymond, and J. Teruyama. Quantum counterfeit coin problems. *Theoretical Computer Science*, 456:51–64, 2012.

[36] A. Kitaev. Quantum measurements and the Abelian stabilizer problem. 1995.

[37] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[38] T. Lee, R. Mittal, B. W. Reichardt, R. Špalek, and M. Szegedy. Quantum query complexity of state conversion. In *Proc. of 52nd IEEE FOCS*, pages 344–353, 2011.

[39] A. Montanaro and R. de Wolf. A survey of quantum property testing. *Theory of Computing*, 2015. To appear.

[40] S. Raskhodnikova, D. Ron, A. Shpilka, and A. Smith. Strong lower bounds for approximating distribution support size and the distinct elements problem. *SIAM Journal on Computing*, 39(3):813–842, 2009.

[41] B. W. Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function. In *Proc. of 50th IEEE FOCS*, pages 544–551, 2009.

[42] B. W. Reichardt. Reflections for quantum query algorithms. In *Proc. of 22nd ACM-SIAM SODA*, pages 560–569, 2011.

[43] B. W. Reichardt and R. Špalek. Span-program-based quantum algorithm for evaluating formulas. *Theory of Computing*, 8:291–319, 2012.

[44] B. E. Sagan. *The symmetric group: representations, combinatorial algorithms, and symmetric functions*, volume 203 of *Graduate Texts in Mathematics*. Springer, 2001.

[45] A. A. Sherstov. The communication complexity of gap Hamming distance. *Theory of Computing*, 8(8):197–208, 2012.

[46] W. van Dam and I. E. Shparlinski. Classical and quantum algorithms for exponential congruences. In *Proc. of 3rd TQC*, volume 5106 of *LNCS*, pages 1–10. Springer, 2008.

[47] M. Zhandry. How to construct quantum random functions. In *Proc. of 53rd IEEE FOCS*, pages 679–687, 2012.

[48] M. Zhandry. A note on the quantum collision and set equality problems. *Quantum Information & Computation*, 15(7&8):557–567, 2015.