Robust Polynomials and Quantum Algorithms

Harry Buhrman[1,2⋆], Ilan Newman[3⋆⋆], Hein Röhrig[4], and Ronald de Wolf[1]

[1] CWI, Amsterdam, the Netherlands.
[2] ILLC, University of Amsterdam, the Netherlands.
[3] Dept. of Computer Science, Haifa University, Israel.
[4] Dept. of Computer Science, University of Calgary, Canada.

Abstract. We define and study the complexity of *robust* polynomials for Boolean functions and the related fault-tolerant quantum decision trees, where input bits are perturbed by noise. We show that, in contrast to the classical model of Feige et al., every Boolean function can be computed by $O(n)$ quantum queries even in the model with noise. This implies, for instance, the somewhat surprising result that every Boolean function has robust degree bounded by $O(n)$.

1 Introduction

In the last two decades, polynomials of many varieties have been used to good effect in complexity theory. We study a variety here that is tailored to analyzing algorithms with *noisy input*.

Robust Polynomials. A *robust* polynomial for a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is a real multivariate polynomial $p(z_1, \ldots, z_n)$ such that for every $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ and every $z = (z_1, \ldots, z_n) \in \mathbb{R}^n$, if $\forall i : |x_i - z_i| \leq 1/3$ then $|f(x) - p(z)| \leq 1/3$ (the $1/3$ in both cases can be changed to any other constant). The *robust degree* of f is the smallest degree of a robust polynomial for f; note that we do not require robust polynomials to be multilinear.

The motivation behind the definition of robust polynomials is twofold: First it can be viewed as a strengthening (restriction) of the notion of approximating polynomials. An approximating polynomial for f is a multivariate real polynomial q that approximates f within an additive term of $1/3$ for each Boolean input. Approximating polynomials for Boolean functions are of interest in themselves and have been the object of study for a while. Their minimal degree is tightly related to the decision tree complexity of f [9, 2]. Indeed, this "polynomial method" [2] is one of the main tools for obtaining lower bounds on the number of queries in quantum algorithms. One difficulty, however, is that approximating polynomials do not directly compose; if $f(x_1, \ldots, x_n)$ is a Boolean function with an approximating polynomial p_f and $g(y_1, \ldots, y_m)$ is a Boolean function

⋆ H.B., H.R., and R.d.W. supported in part by the EU fifth framework projects QAIP, IST-1999-11234, and RESQ, IST-2001-37559, and an NWO grant.
⋆⋆ partially supported by ISF grant 55/03

with an approximating polynomial p_g, then the polynomial on $n \cdot m$ variables $p_f(p_g, \ldots, p_g)$, which is obtained by plugging in a copy of p_g for each appearance of y_i, is not necessarily an approximating polynomial for the composed function $f(g, \ldots, g)$ on $n \cdot m$ variables. This difficulty is avoided with robust polynomials; if p_f, p_g are robust for f, g respectively, then their composition is a robust polynomial (and thus also approximating) for the composed function.

Another motivation is the study of quantum decision trees that can tolerate noise in their inputs. We show that a natural quantum analogue of classical fault-tolerant decision trees can be defined. As a result, it will follow that every such algorithm (and in fact every classical noisy decision tree algorithm as well) implies the existence of a robust degree-$2q$ polynomial for the function, where q is the number of queries. This relates the robust degree to fault-tolerant computation in exactly the same way that approximating polynomials are related to bounded-error quantum algorithms. Surprisingly, our results imply robust quantum algorithms with a linear number of queries, as well as robust polynomials of linear degree, for *any* Boolean function. This should be contrasted with the result of Feige et al. [3] who proved that for most Boolean functions an overhead factor of $\Omega(\log n)$ on the number of queries is needed in the noisy case compared to the non-noisy case. In particular, consider the parity function on n variables. This function can be decided trivially by an n-query decision tree, and hence can be represented exactly by an n-degree real multilinear polynomial (which is just the single monomial containing all variables in the $\{-1, 1\}$ representation). Feige et al. [3] prove that in the noisy decision tree any algorithm for PARITY needs $\Omega(n \log n)$ queries. Using standard amplification techniques, this yields a $\mathrm{O}(n \log n)$-degree robust polynomial for PARITY. Can one do better? Our results imply that there is a robust polynomial for PARITY of degree $\mathrm{O}(n)$. However, we only have an indirect description of this polynomial by means of a quantum algorithm, and do not know how to construct such a polynomial directly.

Noisy Quantum Queries. We now discuss in more detail the model of noisy-decision trees in the quantum world. The notion of a "noisy query" in the quantum case is not obvious and natural as in the classical case since one application of a quantum query can address many different x_i's in superposition. A first proposal would be that for each quantum query, each of the bits is flipped independently with probability ϵ. Each such quantum query introduces a lot of randomness and the algorithm's state after the query is a mixed quantum state rather than a pure state. In fact, this model is a concrete (and very destructive) form of decoherence; the effects of various forms of decoherence on oracle algorithms like Grover's have been studied before, see e.g., [8, 10].

A second model, which we will adopt here, is to assume that we have n quantum procedures, $A_1, \ldots, A_n$, such that A_i outputs x_i with probability at least $1 - \epsilon$. Such a *coherent-noise model* is not unreasonable. For instance, it could be the case that the input bits are actually computed for us by subroutines. Such algorithms can always be made coherent by pushing measurements to the end, which means that we can apply and reverse them at will. To enable us

to apply the A_i's in superposition, we assume we have a black box that maps $\mathcal{A} : |i\rangle|0\rangle \mapsto |i\rangle A_i|0\rangle$. One application of this will count as one query.

A third model, which we will call the *multiple-noisy-copies model*, was studied by Szegedy and Chen [11]. Here, instead of x_i, the algorithm can only query "perturbed" copies $y_{i,1}, \ldots, y_{i,m}$ of x_i. The $y_{i,j}$ are independent Boolean random variables with $\Pr[x_i = y_{i,j}] \geq 1 - \epsilon$ for each $i = 1, \ldots, n$ and $j = 1, \ldots, m$. In contrast to the first proposal, this model leaves the queries perfectly reversible, since the perturbed copies are fixed at the start of the algorithm and the same $y_{i,j}$ can be queried more than once. The assumption of this model is also stronger than the second model, since we can construct a 1-query A_i that just outputs a superposition of all $y_{i,j}$. If m is sufficiently large, this A_i will compute x_i with high success probability, satisfying the assumption of the second model (see Section 4.2 for details).

Robust Quantum Algorithms. Assuming the second model and some fixed ϵ, we call a quantum algorithm *robust* if it computes f with bounded error probability on inputs of n bits given by bounded-error algorithms $A_1, \ldots, A_n$, respectively. A first observation is that every T-query non-robust algorithm can be made robust at a multiplicative cost of $\mathrm{O}(\log T)$. With $\mathrm{O}(\log T)$ queries, a majority gate, and an uncomputation step, we can construct a unitary $\tilde{U}_x$ that approximates an exact quantum query $U_x : |i\rangle|b\rangle \mapsto |i\rangle|b \oplus x_i\rangle$ very well: $\|U_x - \tilde{U}_x\| \leq 1/100T$. Since errors add linearly in a quantum algorithm, replacing U_x by $\tilde{U}_x$ in a non-robust algorithm gives a robust algorithm with almost the same final state. In some cases better constructions are possible. For instance, a recent result by Høyer et al. [5] implies a quantum algorithm that robustly computes Or with $\mathrm{O}(\sqrt{n})$ queries. This is only a constant factor worse than the noiseless case, which is Grover's algorithm [4]. In fact, we do not know of any function where the robust quantum query complexity is more than a constant factor larger than the non-robust complexity.

Our main result about quantum computing (made precise in Theorem 2) is the following:

> There exists a quantum algorithm that outputs $x_1, \ldots, x_n$, with high probability, using $\mathrm{O}(n)$ invocations of the A_i algorithms (i.e., queries).

As already mentioned, this result implies that *every* n-bit function f can be robustly quantum computed with $\mathrm{O}(n)$ queries. This contrasts with the classical $\Omega(n \log n)$ lower bound for Parity. It is quite interesting to note that quantum computers, which usually are more fragile than classical computers, are actually more robust in the case of computing Parity with noisy inputs. The result for Parity can be extended to every symmetric function f: for every such function, the optimal quantum algorithm can be made robust with only a constant factor overhead (see Section 4.1).

Our result has a direct bearing on the *direct-sum problem*, which is the question how the complexity of computing n independent instances of a function scales with the complexity of one instance. One would expect that computing n instances with bounded-error takes no more than n times the complexity of one

instance. However, since we want all n instances to be computed correctly *simultaneously* with high probability, the only known general method in the classical world is to compute each instance with error probability reduced to $O(1/n)$. This costs another factor of $O(\log n)$. In fact, it follows from the $\Omega(n \log n)$ bound for PARITY that this factor of $\log n$ is optimal if we can only run algorithms for individual instances in a black-box fashion. In contrast, our result implies that in the quantum world, the bounded-error complexity of n instances is at most $O(n)$ times the bounded-error complexity of one instance. This is a very general result. For example, it also applies to communication complexity [7, Section 4.1.1]. If Alice and Bob have a bounded-error protocol for a distributed function f, using c bits (or qubits) of communication, then there is a bounded-error quantum protocol for n instances of f, using $O(n(c + \log n))$ qubits of communication. The additive $\log n$ is because Alice and Bob need to communicate (possibly in superposition) the index of the instance that they are computing. In contrast, the best known general classical solution uses $\Theta(cn \log n)$ bits of communication.

Note about Related Work. In a recent manuscript, Iwama et al. [6] study a similar but slightly weaker setting. There, the error probability for each input variable is *exactly* ϵ. If ϵ is known, then one can use a version of exact amplitude amplification to "rotate off" the error using $O(1)$ queries and hence make the algorithm robust. If ϵ unknown, it can be estimated very well using quantum amplitude estimation, after which amplitude amplification can be used as if ϵ was known. Iwama et al. derive from this that any quantum algorithm can be made robust (in their model) with only a constant factor overhead. Their model has the disadvantage that it does not cover the subroutine-scenario, where each input bit x_i is computed for us by an algorithm or subroutine A_i whose error we can only upper bound. Our model does not need the assumption that the error is the same for all input bits, and hence does not have this disadvantage.

2 Robust Polynomials — Preliminaries

In this section we study robust polynomials of two different but essentially equivalent types. The first type arises from the multiple-noisy-copies model, the second type is what we discussed in the introduction. For brevity, we omit the proofs of the lemmas presented in this section.

Definition 1. *An* (ϵ, m) perturbation *of* $x \in \{0,1\}^n$ *is a matrix* y *of* $n \times m$ *independent binary random variables* $y_{i,j}$ *so that* $\Pr[y_{i,j} = x_i] \geq 1 - \epsilon$ *for each* $1 \leq j \leq m$.

Definition 2. *A* type-1 (ϵ, m)-robust polynomial *for the Boolean function* $f : \{0,1\}^n \to \{0,1\}$ *is a real polynomial* p *in* nm *variables* $y_{i,j}$ *(with* $1 \leq i \leq n$ *and* $1 \leq j \leq m$*) so that for every* $x \in \{0,1\}^n$ *and* y *an* (ϵ, m) *perturbation of* x, $\Pr[|p(y) - f(x)| \geq 1/3] \leq 1/3$. *Moreover, for every* $v \in \{0,1\}^{nm}$, *we require* $-1/3 \leq p(v) \leq 4/3$.

Note that since $y_{i,j}^2 = y_{i,j}$ for a bit $y_{i,j}$, we can restrict attention to *multilinear* polynomials here.

The approximation "quality" of a type-1 robust polynomial can be boosted at constant multiplicative cost in the degree. Analogously we can improve the parameters to other constants:

Lemma 1. *If there is a* type-1 (ϵ, m)-robust polynomial *of degree d for f, then for some $m' = \mathrm{O}(m)$ there exists a type-1 (ϵ, m')-robust polynomial p of degree $\mathrm{O}(d)$ so that $x \in \{0,1\}^n$ and y an (ϵ, m') perturbation of x, $\Pr[|p(y) - f(x)| \geq 1/9] \leq 1/9$. Moreover, for any $v \in \{0,1\}^{nm'}$, $-1/9 \leq p(v) \leq 10/9$.*

The second kind of robust polynomial is the following:

Definition 3. *For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, we call q a* type-2 ϵ-robust polynomial for f *if q is a real polynomial in n variables so that for every $x \in \{0,1\}^n$ and every $z \in [0,1]^n$ we have $|q(z) - f(x)| \leq 1/3$ if $|z_i - x_i| \leq \epsilon$ for all $i \in [n]$. If $\epsilon = 0$, then q is called an* approximating polynomial *for f.*

A minimal-degree type-2 robust polynomial for f need not be multilinear, in contrast to the type-1 variety. Note that we restrict the z_i's to lie in the set $[0,\epsilon] \cup [1-\epsilon, 1]$ rather than the less restrictive $[-\epsilon, \epsilon] \cup [1-\epsilon, 1+\epsilon]$. This facilitates later proofs, because it enables us to interpret the z_i's as probabilities. However, with some extra work we could also use the less restrictive definition here.

Definition 4. *For $f : \{0,1\}^n \to \{0,1\}$, let $\mathrm{rdeg}_1(f)$ denote the minimum degree of any type-1 $(1/3, 5\log n)$-robust polynomial for f, $\mathrm{rdeg}_2(f)$ be the minimum degree of any type-2 $1/3$-robust polynomial approximating f, and $\widetilde{\deg}(f)$ be the minimum degree among all approximating polynomials for f.*

We characterize the relation of type-1 and type-2 robust polynomials as follows:

Theorem 1. *For every type-2 ϵ-robust polynomial of degree d for f there is a type-1 $(\epsilon/2, \mathrm{O}(\log(n)/(1/2-\epsilon)^2))$-robust polynomial of degree d for f. Conversely, for every type-1 (ϵ, m)-robust polynomial of degree d for f there is a type-2 ϵ-robust polynomial of degree $\mathrm{O}(d)$ for f.*

Proof. Let p be a type-2 ϵ-robust polynomial of degree d for f. We choose $m = \mathrm{O}(\log(n)/(1/2-\epsilon)^2)$. If each $y_{i,j}$ is wrong with probability $\leq \epsilon/2$, then with probability at least $2/3$, the averages $\overline{y}_i$ will satisfy $|\overline{y}_i - x_i| \leq \epsilon$ for all $i \in [n]$. Hence the polynomial $p(\overline{y}_1, \ldots, \overline{y}_n)$ will be a type-1 $(\epsilon/2, \mathrm{O}(\log(n)/(1/2-\epsilon)^2))$-robust polynomial of degree d for f.

For the other direction, consider a type-1 (ϵ, m)-robust polynomial of degree d for f. Using Lemma 1, we boost the approximation parameters to obtain a type-1 (ϵ, m')-robust polynomial p of degree $\mathrm{O}(d)$, with $m' = \mathrm{O}(m)$, so that for any $x \in \{0,1\}^n$ and (ϵ, m') perturbation y of x, $\Pr[|p(y) - f(x)| \geq 1/9] \leq 1/9$. For $z \in \mathbb{R}^n$ with $0 \leq z_i \leq 1$ for all i, let $y_{i,j}$ ($i \in [n]$, $j \in [m']$) be independent random variables, where $y_{i,j} = 1$ with probability z_i. Define $q(z) := \mathrm{E}[p(y)]$. This q is a polynomial in z, because $\mathrm{E}[p(y)] = p(\mathrm{E}[y])$ and $\mathrm{E}[y_{i,j}] = z_i$. Moreover, if for z there exists $x \in \{0,1\}^n$ with $|z_i - x_i| \leq \epsilon$ for all i, then y is an (ϵ, m')

perturbation of x. Therefore $V := \{v : |p(v) - f(x)| \leq 1/9\}$ has probability $\Pr[y \in V] \geq 8/9$ and

$$|f(x) - q(z)| \leq \left|\sum_{v \in V} \Pr[y = v]\,(f(x) - p(v))\right| + \left|\sum_{v \notin V} \Pr[y = v]\left(1 + \frac{1}{9}\right)\right| < \frac{1}{3}\ .$$

This means that $q(z)$ is a type-2 ϵ-robust polynomial for f of degree $\mathrm{O}(d)$. □

Note that in Definition 2 we require for type-1 polynomials p that for any Boolean assignment $v \in \{0,1\}^{nm}$ to the (possibly real) variables, the polynomial value $p(v)$ lies between $-1/3$ and $4/3$. Because of this totality requirement, the following corollary is given for total Boolean f only.

Corollary 1. $\mathrm{rdeg}_1(f) = \Theta(\mathrm{rdeg}_2(f))$ *for every (total) Boolean function* $f : \{0,1\}^n \to \{0,1\}$.

Robust quantum algorithms provide one way to construct robust polynomials:

Lemma 2. *Let* $f : \{0,1\}^n \to \{0,1\}$ *be a Boolean function. Let* Q *be a quantum algorithm that makes at most* q *queries on inputs from* $\{0,1\}^{n \times m}$. *If for every* $x \in \{0,1\}^n$ *and* y *an* (ϵ, m) *perturbation of* x, *we have that* $\Pr[Q(y) = f(x)] \geq 29/30$, *then there exists a degree-*$2q$ *type-1* (ϵ, m)*-robust polynomial for* f.

3 Quantum Robust Input Recovery

In this section we prove our main result, that we can recover an n-bit string x using $\mathrm{O}(n)$ invocations of algorithms $A_1, \ldots, A_n$ where A_i computes x_i with bounded error.

Theorem 2. *Given* ϵ*-error algorithms* $A_1, \ldots, A_n$ *for the bits* $x_1, \ldots, x_n$, *there is a quantum algorithm that recovers* $x = x_1 \ldots x_n$ *with probability* $2/3$ *using* $\mathrm{O}(n/(1/2 - \epsilon)^2)$ *queries (invocations of the* A_i*).*

We assume that $\epsilon > 0$ and that A_i is a unitary transformation

$$A_i : |0^t\rangle \mapsto \alpha_i |0\rangle|\psi_i^0\rangle + \sqrt{1 - \alpha_i^2}\,|1\rangle|\psi_i^1\rangle$$

for some $\alpha_i \geq 0$ such that $|\alpha_i|^2 \leq \epsilon$ if $x_i = 1$ and $|\alpha_i|^2 \geq 1 - \epsilon$ if $x_i = 0$; $|\psi_i^0\rangle$ and $|\psi_i^1\rangle$ are arbitrary $(t-1)$-qubit norm-1 quantum states. It is standard that any quantum algorithm can be expressed in this form by postponing measurements (i.e., unitarily write the measurement in an auxiliary register without collapsing the state); any classical randomized algorithm can be converted into this form by making it reversible and replacing random bits by states $(|0\rangle + |1\rangle)/\sqrt{2}$. By applying a NOT to the first qubit after the execution of A_i, we can easily implement

$$\bar{A}_i : |0^t\rangle \mapsto \alpha_i |1\rangle|\psi_i^0\rangle + \sqrt{1 - \alpha_i^2}\,|0\rangle|\psi_i^1\rangle\ ,$$

Procedure RobustFind(n, $\mathcal{A}$, ϵ, β, γ, δ)

$n \in \mathbb{N}$, $\mathcal{A} : n$ quantum algorithms, $\epsilon, \beta, \gamma, \delta > 0$
Output: $i \in [n] \cup \{\perp\}$ with the following properties:

1. if $\mathcal{A}$ is ϵ-close to $x \in \{0,1\}^n$ and $|x| \geq \beta n$, then $i \neq \perp$ with probability $\geq 1 - \delta$
2. if $\mathcal{A}$ is ϵ-close to $x \in \{0,1\}^n$ and if $i \neq \perp$, then $x_i = 1$ with probability $\geq 1 - \gamma$

Complexity: $\mathrm{O}\left(\frac{1}{(\frac{1}{2}-\epsilon)^2} \cdot \sqrt{\frac{1}{\beta}} \cdot \log \frac{1}{\gamma\delta}\right)$ invocations of the A_i

Procedure AllInputs(n, $\mathcal{A}$, ϵ)

$n \in \mathbb{N}$, $\mathcal{A} : n$ algorithms, $\epsilon > 0$

1: **for** $i \leftarrow$ to n **do**
2: run A_i
3: $\tilde{x}_i \leftarrow$ result of A_i
4: **for** $k \leftarrow 1$ to $\log(\epsilon(\log n)^2)$ **do**
5: $\epsilon' \leftarrow \epsilon/2^{k-1}$
6: **for** $\ell \leftarrow 1$ to $1.7\epsilon' n$ **do**
7: $i \leftarrow \text{RobustFind}(n, \mathcal{A}(\tilde{x}), \epsilon, 0.3\epsilon', \frac{1}{8}, \frac{1}{8})$
8: **if** $i \neq \perp$ **then**
9: $\tilde{x}_i \leftarrow 1 - \tilde{x}_i$
10: **for** $m \leftarrow n/(\log n)^2$ down to 1 **do**
11: $i \leftarrow \text{RobustFind}(n, \mathcal{A}(\tilde{x}), \epsilon, \frac{m}{n}, \frac{1}{20n}, \frac{1}{20n})$
12: **if** $i \neq \perp$ **then**
13: $\tilde{x}_i \leftarrow 1 - \tilde{x}_i$
14: **return** $\tilde{x}$

which operates like A_i but outputs 1 when A_i would have output 0 and vice versa. Define $A_i(b)$ by $A_i(0) := A_i$ and $A_i(1) = \bar{A}_i$. If we plug the right bit x_i into A_i, then for all A_i we expect output 0: for the unique good $x \in \{0,1\}^n$, $\mathcal{A}(x) := (A_1(x_1), \ldots, A_n(x_n))$ is ϵ-close to 0^n by the following notion of closeness:

Definition 5. *For $\epsilon < 1/2$ and decision algorithms $\mathcal{A} = (A_1, \ldots, A_n)$, we say $\mathcal{A}$ is* ϵ-close *to $x \in \{0,1\}^n$ if $\Pr[A_i$ outputs $x_i] \geq 1 - \epsilon$ for all $i \in [n]$.*

Our algorithm builds on a robust quantum search algorithm by Høyer, Mosca, and de Wolf [5], which we call RobustFind. This subroutine takes a vector $\mathcal{A}$ of n quantum algorithms and in the good case returns an index i so that the "high probability" output of A_i is 1. This allows us to verify a purported solution $\tilde{x} \in \{0,1\}^n$ by running RobustFind on $\mathcal{A}(\tilde{x})$ to find differences with the real input x. In fact, adjusting the parameters to RobustFind as we move closer and closer to a good solution, AllInputs (as defined by the pseudo code on page 7) manages to construct the unique x with high probability. Note that RobustFind is the only quantum component of our otherwise classical algorithm.

The first step of our algorithm (Lines 1–3) is to classically sample each i once and to store this initial approximation into a variable $\tilde{x}_i$. We call $i \in [n]$ a *bad* index if $x_i \neq \tilde{x}_i$. The following rounds of the algorithm (Lines 4–9) use

RobustFind with error probabilities $\gamma = \delta = 1/8$ and a decreasing estimate of the number of bad indices, β. This way we refine $\tilde{x}$ until we have fewer than $n/(\log n)^2$ bad indices. At that point, we can afford to set γ and δ to very small values to eliminate all remaining bad indices with high probability.

Success probability. Let B_0 denote the random variable counting the number of bad indices after Line 3 and let B_k denote the random variable of the number of bad indices after the iteration k of the `for` loop in Lines 4–9. By $\mathcal{G}_k$ we denote the event $B_k \leq n\epsilon/2^{k-1}$. We have

$$\Pr[\mathcal{G}_{k_{\max}}] \geq \Pr[\mathcal{G}_0] \prod_{k=1}^{k_{\max}} \Pr[\mathcal{G}_k | \mathcal{G}_{k-1}] \quad . \tag{1}$$

We now show that $\Pr[\mathcal{G}_k|\mathcal{G}_{k-1}]$ is large by means of Chernoff bounds on the number of bad indices that we find in round k and the number of errors we make. For $k = 0$, we know that $\mathrm{E}[B_0] \leq \epsilon n$ and thus $\Pr[B_0 \leq 2\epsilon n] \geq 0.9$. In round k, we want to reduce the upper bound on the number of bad indices from $2n\epsilon/2^{k-1}$ to $n\epsilon/2^{k-1}$. Let E_k denote the random variable of the number of errors that we make in round k, i.e., the number of wrongly identified bad indices. Similarly, let C_k denote the random variable of the number of correctly identified bad indices. Then $B_k = B_{k-1} - C_k + E_k$, so

$$\Pr[\mathcal{G}_k|\mathcal{G}_{k-1}] \geq \Pr[E_k \leq 1.1\gamma r \mid \mathcal{G}_{k-1}] \cdot \Pr\left[C_k \geq B_{k-1} - \frac{n\epsilon}{2^{k-1}} + 1.1\gamma r \mid \mathcal{G}_{k-1}\right] \tag{2}$$

We choose the number of repetitions of RobustFind to be $r := 1.7n\epsilon/2^{k-1}$ and our lower bound on the fraction of bad indices to be $\beta := 0.3\epsilon/2^{k-1}$. Then $\mathrm{E}[E_k|\mathcal{G}_{k-1}] \leq \gamma r$ and $\Pr[E_k \leq 1.1\gamma r \mid \mathcal{G}_{k-1}] \geq 1 - e^{-\Omega(r)}$. To bound the second factor in (2), we need to take into account that we have no guarantee on the success probability of a single RobustFind invocation if the number of bad indices falls below βn. However, if this happens, then $C_k \geq B_{k-1} - \beta n \geq B_{k-1} - n\epsilon/2^{k-1} + 1.1\gamma r$, i.e., the second factor in (2) is trivially 1. Therefore it is safe to assume that each invocation of RobustFind has success probability at least $(1-\gamma)(1-\delta)$. Hence,

$$\mathrm{E}[B_{k-1} - C_k + 1.1\gamma r \mid \mathcal{G}_{k-1}] \leq 2\frac{n\epsilon}{2^{k-1}} - (1-\gamma)(1-\delta)r + 1.1\gamma r \leq 0.9\frac{n\epsilon}{2^{k-1}}$$
$$\text{and } \Pr\left[B_{k-1} - C_k + 1.1\gamma r \leq \frac{n\epsilon}{2^{k-1}} \mid \mathcal{G}_{k-1}\right] \geq 1 - e^{-\Omega(r)} \quad .$$

Altogether, this establishes $\Pr[\mathcal{G}_k|\mathcal{G}_{k-1}] \geq 1 - e^{-\Omega(r)}$.

Substituting this bound into (1) we obtain, with $k_{\max} = \log(\epsilon(\log n)^2)$ and $r = 1.7n\epsilon/2^{k-1} = \Omega(n/(\log n)^2)$,

$$\Pr[\mathcal{G}_{k_{\max}}] \geq \Pr[\mathcal{G}_0]\left(1 - e^{-\Omega(r)}\right)^{k_{\max}} \geq 0.9\left(1 - \frac{\log(\epsilon(\log n)^2)}{e^{\Omega(n/(\log n)^2)}}\right) = 0.9 - \mathrm{o}(1).$$

Hence, for large n with probability 0.8 we have at most $n/(\log n)^2$ bad indices at the end of the `for` loop in Lines 4–9. In this case, we will find with constant probability all bad indices by making the individual error probability in

RobustFind so small that we can use a union bound: we determine each of the remaining bad indices with error probability $1/(10n)$. This implies an overall success probability $\geq 0.8 \cdot 0.9 > 2/3$.

Complexity. We bound the number of queries to f in Lines 4–9 as follows:

$$\sum_{k=1}^{k_{\max}} \sum_{\ell=1}^{n\epsilon/2^{k-1}} C \frac{1}{\left(\frac{1}{2}-\epsilon\right)^2} \sqrt{\frac{1}{\epsilon/2^k}} \leq C' \frac{\sqrt{\epsilon}}{\left(\frac{1}{2}-\epsilon\right)^2} n \sum_{k=1}^{\infty} \frac{1}{2^{k/2}} = \mathrm{O}\left(\frac{n}{\left(\frac{1}{2}-\epsilon\right)^2}\right)$$

for some constants C, C'. Lines 10–13 result in

$$\mathrm{O}\left(\sum_{m=1}^{n/(\log n)^2} \frac{1}{\left(\frac{1}{2}-\epsilon\right)^2} \sqrt{\frac{n}{m}} \log n\right) = \mathrm{O}\left(\frac{n}{\left(\frac{1}{2}-\epsilon\right)^2}\right)$$

many queries. Therefore, the total query complexity of AllInputs is $\mathrm{O}(n/(1/2-\epsilon)^2)$.

4 Making Quantum Algorithms Robust

4.1 Inputs Computed by Quantum Algorithms

Here we state a few corollaries of Theorem 2. First, once we have recovered the input x we can compute any function of x without further queries, hence

Corollary 2. *For every $f : \{0,1\}^n \to \{0,1\}$, there is a robust quantum algorithm that computes f using $\mathrm{O}(n)$ queries.*

In particular, PARITY can be robustly quantum computed with $\mathrm{O}(n)$ queries while it takes $\Omega(n \log n)$ queries classically [3].

Second, in the context of the direct-sum problem, the complexity of quantum computing a vector of instances of a function scales linearly with the complexity of one instance.

Corollary 3 (Direct Sum). *If there exists a T-query bounded-error quantum algorithm for f, then there is an $\mathrm{O}(Tn)$-query bounded-error quantum algorithm for n independent instances of f.*

As mentioned, the best classical upper bound has an additional factor of $\log n$, and this is optimal in a classical black-box setting.

Thirdly, all *symmetric* functions can be computed robustly on a quantum computer with the same asymptotic complexity as non-robustly. A function is symmetric if its value only depends on the Hamming weight of the input. Let $\Gamma(f) := \min\{|2k-n+1| : f \text{ changes value if the Hamming weight of the input changes from } k \text{ to } k+1\}$. The non-robust algorithm for computing f with $\mathrm{O}(\sqrt{n(n-\Gamma(f))})$ queries [1, Theorem 4.10] can be made robust by a similar algorithm as the one used in the proof of our Theorem 2, giving:

Theorem 3. *For every symmetric function f, there is a robust quantum algorithm that computes f using $\mathrm{O}(\sqrt{n(n-\Gamma(f))})$ quantum queries.*

4.2 Multiple Noisy Copies

As mentioned in the introduction, the assumption that we have a bounded-error algorithm A_i for each of the input bits x_i also covers the model of [11] where we have a sequence $y_{i,1}, \ldots, y_{i,m}$ of noisy copies of x_i. These we can query by means of a mapping $|i\rangle|j\rangle|0\rangle \mapsto |i\rangle|j\rangle|y_{i,j}\rangle$. Here we spell out this connection in some more detail. First, by a Chernoff bound, choosing $m := \mathrm{O}(\log(n)/\epsilon^2)$ implies that the average $\overline{y}_i := \sum_{j=1}^{m} y_{i,j}/m$ is close to x_i with very high probability: $\Pr[|\overline{y}_i - x_i| \geq 2\epsilon] \leq 1/(100n)$. By the union bound, with probability 99/100 this closeness will hold for all $i \in [n]$ simultaneously. Assuming this is the case, we implement the following unitary mapping using one query: $A_i : |0^{\log(m)+1}\rangle \mapsto \frac{1}{\sqrt{m}} \sum_{j=1}^{m} |j\rangle|y_{i,j}\rangle$. Measuring the last qubit of the resulting state gives x_i with probability at least $1 - 2\epsilon$. Hence, we can run our algorithm from Section 3 and recover x using $\mathrm{O}(n)$ queries to the $y_{i,j}$. Similarly, all consequences mentioned in Section 4.1 hold for this multiple-noisy-copies model as well.

5 Making Approximating Polynomials Robust

Theorem 4. $\mathrm{rdeg}_{1,2}(f) = \mathrm{O}(n)$ *for every* $f : \{0,1\}^n \to \{0,1\}$.

Proof. By Corollary 2 and the discussion in Section 4.2, f has an $\mathrm{O}(n)$-query robust quantum algorithm in the multiple-noisy-copies model that operates on $\mathrm{O}(\log n)$ copies. By Lemma 2 this induces a type-1 robust polynomial for f of degree $\mathrm{O}(n)$. And finally, by Corollary 1 there also exists a degree-$\mathrm{O}(n)$ type-2 robust polynomial for f. □

In particular, this shows that for functions with approximate degree $\Theta(n)$ we can make the approximating polynomial robust at only constant factor overhead in the degree. This case includes explicit functions like Parity and Majority, but also random (hence almost all) functions. It is open whether approximating polynomials can *always* be made robust at only a constant overhead in the degree. The best we can do is show that a non-robust degree-d approximating polynomial can be made robust at a cost of a factor $\mathrm{O}(\log d)$. Our proof makes use of the well known notion of *certificate complexity*.

Definition 6. *An assignment* $C : S \to \{0,1\}$ *of values to some subset* $S \subseteq [n]$ *of the* n *variables is* consistent *with* $x \in \{0,1\}^n$ *if* $x_i = C(i)$ *for all* $i \in S$. *For* $b \in \{0,1\}$, *a* b-certificate *for* f *is an assignment* C *such that* $f(x) = b$ *whenever* x *is consistent with* C. *The* size *of* C *is* $|S|$, *the cardinality of* S. *The* certificate complexity $C_x(f)$ of f on x *is the size of a smallest* $f(x)$*-certificate that is consistent with* x. *The* certificate complexity *of* f *is* $C(f) = \max_x C_x(f)$.

Lemma 3. *Let* p *be an* ϵ*-approximating polynomial for* $f : \{0,1\}^n \to \{0,1\}$, *and* $c = C(f)$ *be the certificate complexity of* f. *If* $x \in \{0,1\}^n$ *and* $z \in [0,1]^n$ *satisfy* $|x_i - z_i| \leq 1/10c$ *for all* $i \in [n]$, *then* $|p(z) - f(x)| \leq \epsilon + 2/15$.

Proof. Consider a certificate C for x of size c. We will use x^C and $x^{\overline{C}}$ to denote the parts of x corresponding to C and to its complement, respectively, and write $x = x^C x^{\overline{C}}$. If $y \in \{0,1\}^n$ is chosen according to the z-distribution ($y_i = 1$ with probability z_i), then

$$p(z) = \mathrm{E}_y[p(y)] = \sum_{y^C y^{\overline{C}}} \Pr[y^C] \Pr[y^{\overline{C}}] p(y^C y^{\overline{C}}) = \sum_{y^{\overline{C}}} \Pr[y^{\overline{C}}] \cdot \mathrm{E}_{y^C}[p(y^C y^{\overline{C}})] \ .$$

Now consider the expectation $\mathrm{E}_{y^C}[p(y^C y^{\overline{C}})]$, where $y^{\overline{C}} \in \{0,1\}^{n-c}$ is fixed, while the y^C-bits are still chosen according to the z-distribution. Consider the c-variate polynomial obtained from p by fixing the bits in $y^{\overline{C}}$. Since the "error" in the z^C-variables is at most $1/10c$, we have $\Pr[y^C = x^C] \geq (1 - 1/10c)^c \geq 9/10$, so $|\mathrm{E}_{y^C}[p(y^C y^{\overline{C}})] - p(x^C y^{\overline{C}})| \leq (1/10)(4/3) = 2/15$. But $f(x^C y^{\overline{C}}) = f(x)$, because the input $x^C y^{\overline{C}}$ satisfies the same certificate as x. Hence

$$|\mathrm{E}_{y^C}[p(y^C y^{\overline{C}})] - f(x)| \leq |\mathrm{E}_{y^C}[p(y^C y^{\overline{C}})] - p(x^C y^{\overline{C}})| + |p(x^C y^{\overline{C}}) - f(x)| \leq 2/15 + \epsilon,$$

and also $|p(z) - f(x)| \leq \epsilon + 2/15$. □

This lemma implies that we can make a non-robust approximating polynomial robust at the cost of a factor of $\mathrm{O}(\log C(f))$ in the degree (replace each variable by a $\mathrm{O}(\log C(f))$-degree error-reducing polynomial). Since $C(f)$ and $\widetilde{\deg}(f)$ are polynomially related ($C(f) = \mathrm{O}(\widetilde{\deg}(f)^4)$, see [2]), we obtain:

Theorem 5. $\mathrm{rdeg}_{1,2}(f) = \mathrm{O}(\widetilde{\deg}(f) \cdot \log \widetilde{\deg}(f))$.

6 Summary and Open Problems

The main results of this paper are as follows:

- For every n-bit Boolean function f there is an n-variate polynomial p of degree $\mathrm{O}(n)$ that *robustly* approximates it, i.e., $p(x)$ remains close to $f(x)$ if we slightly vary the n inputs.
- There is an $\mathrm{O}(n)$-query quantum algorithm that *robustly* recovers n noisy input bits. Hence every n-bit function can be quantum computed with $\mathrm{O}(n)$ queries in the presence of noise. This contrasts with the classical case, where most functions need $\Theta(n \log n)$ queries.

Note that the use of the robust Or algorithm by [5] is not necessary for recovering the whole input. It would suffice to use Grover's algorithm, that runs in time $\sqrt{n/t}$ when there are t marked items. When we know an estimate of t, like in our $\log n$ rounds algorithm, we can make the error of a single query as small as $1/\sqrt{n/t}$ at the cost of a $\log(n/t)$ multiplicative factor. Standard analysis shows that in this case Grover's algorithm behaves as the robust Or algorithm. However this way we would not obtain the results about every symmetric function (Theorem 3).

We mention some open problems. First, in contrast to the classical case (PARITY) we do not know of any function where making a quantum algorithm robust costs more than a constant factor. Such a constant overhead suffices in the case of symmetric functions and functions whose approximate degree is $\Omega(n)$. It is conceivable that quantum algorithms (and polynomials) can *always* be made robust at a constant factor overhead. Proving or disproving this would be very interesting. We are not aware of a direct "closed form" or other natural way to describe a robust degree-n polynomial for the parity of n bits, but can only infer its existence from the existence of a robust quantum algorithm. Given the simplicity of the non-robust representing polynomial for PARITY, one would hope for a simple closed form for robust polynomials for PARITY as well.

Finally, we have chosen our model of a noisy query such that we can coherently make a query and reverse it. It is not clear to what extent non-robust quantum algorithms can be made resilient against decohering queries, since the usual transformations to achieve fault-tolerant quantum computation do not immediately apply to the query gate, which acts on a non-constant number of quantum bits simultaneously.

Acknowledgments. We thank Peter Høyer for inspiring initial discussions that led to our main result, and Michele Mosca for sending us a version of [6].

References

1. R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. Earlier version in FOCS 98.
2. H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: A survey. *Theoretical Computer Science*, 288(1):21–43, 2002.
3. U. Feige, P. Raghavan, D. Peleg, and E. Upfal. Computing with noisy information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994.
4. L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of 28th ACM STOC*, pages 212–219, 1996.
5. P. Høyer, M. Mosca, and R. de Wolf. Quantum search on bounded-error inputs. In *Proceedings of 30th ICALP*, volume 2719 of *Lecture Notes in Computer Science*, pages 291–299. Springer, 2003.
6. K. Iwama, R. Putra, and S. Yamashita. Quantum query complexity of biased oracles. Unpublished manuscript and talk at EQIS conference, September 2003.
7. E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.
8. G. L. Long, Y. S. Li, W. L. Zhang, and C. C. Tu. Dominant gate imperfection in Grover's quantum search algorithm. *Physical Review A*, 61:042305, 2000.
9. N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, 1994. Earlier version in STOC'92.
10. N. Shenvi, K. R. Brown, and K. B. Whaley. Effects of a random noisy oracle on search algorithm complexity. *Physical Review A*, 68:052313, 2003.
11. M. Szegedy and X. Chen. Computing Boolean functions from multiple faulty copies of input bits. In *Proceedings of 5th LATIN*, volume 2286 of *Lecture Notes in Computer Science*, pages 539–553, 2002.