

File ID 70195
Filename Chapter 4 Robustness

SOURCE (OR PART OF THE FOLLOWING SOURCE):

Type Dissertation
Title Quantum query complexity and distributed computing
Author H.P. Röhrig
Faculty University of Amsterdam
Faculty of Humanities
Faculty of Science
Year 2004
Pages 157
ISBN 3-933966-04-3

FULL BIBLIOGRAPHIC DETAILS:

<http://dare.uva.nl/record/220523>

Copyright

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use.

In this chapter we study the effect of noisy input on problems in the blackbox setting. It is based on work with Buhrman, Newman, and de Wolf [36].

4.1 Introduction

Consider the following setting: we would like to compute some function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, but our access to the input $x \in \{0, 1\}^n$ has to deal with noise: when looking up the bit x_i we get the wrong value $1 - x_i$ with probability ε_i . The precise error probability is unknown to us, but we are given an upper bound $\varepsilon < 1/2$ so that for all bit positions i holds $\varepsilon_i \leq \varepsilon$. Many algorithms designed for noiseless input will fail when given such noisy input. For example, the trivial algorithm for computing OR, “query every bit and output 1 if $x_i = 1$ for at least one i ,” will fail with high probability on the all-zero input for $\varepsilon > 1/n$.

Feige et al. [53] studied the overhead it takes to make an algorithm *robust*, i.e., resistant against noisy inputs. In general, one can query a variable x_i $O(\log n)$ times instead of once and take the majority value as the value of x_i . This reduces the uniform bound on the error probability to much less than $1/n$; then the union bound implies that with high probability *all* queries will be given the correct value, so a non-robust algorithm will work. Accordingly, every non-robust algorithm in the decision-tree or query-complexity model can be made robust at the cost of a factor $O(\log n)$ overhead (in fact, $O(\log T)$ would suffice for a T -query algorithm). Sometimes this factor of $O(\log n)$ is necessary: Feige et al. proved that every robust algorithm for the PARITY function needs to make $\Omega(n \log n)$ queries, for fixed ε . On the other hand, for some functions the $O(\log n)$ can be dispensed with: Feige et al. also designed a non-trivial robust algorithm that computes the OR with $O(n)$ queries, only a constant factor worse than the noiseless case.

Here we study this model for quantum algorithms. There is an issue as to what a “noisy query” means in this case, since one application of a quantum query can address many different x_i ’s in superposition:

1. One possibility is that for each quantum query, each of the bits is flipped with probability ε . However, now each quantum query introduces a lot of randomness, and the algorithm’s state after the query would no longer be a pure quantum state.
2. Alternatively, we can assume that we have n quantum procedures, A_1, \dots, A_n , such that A_i outputs x_i with probability at least $1 - \varepsilon$. 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

$$\mathcal{A} : |i\rangle|0\rangle \mapsto |i\rangle A_i|0\rangle .$$

One application of this will count as one query.

3. The multiple-faulty-copies model was studied by Szegedy and Chen [115]; here, instead of x_i , the algorithm can only query “perturbed” copies $y_{i,1}, \dots, y_{i,m}$ of x_i . The $y_{i,j}$ are independent Boolean random variables with $\Pr[x_i = y_{i,j}] \geq 1 - \varepsilon$ for each $i = 1, \dots, n, j = 1, \dots, m$. 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, A_i will compute x_i with high success probability, satisfying the assumption of the second model (see Section 4.3 for details).

Assuming the second model and some fixed ε , we call a quantum algorithm *robust* if it computes f with bounded error probability when its inputs are given by algorithms A_1, \dots, A_n . A first observation is that every T -query non-robust algorithm can be made robust at a multiplicative cost of $O(\log T)$. With $O(\log T)$ 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. [75] immediately implies a quantum algorithm that robustly computes OR with $O(\sqrt{n})$ queries. This is only a constant factor worse than the noiseless case, which is Grover’s algorithm [69]. In fact, we do not know

of any function where the robust degree is more than a constant factor larger than the non-robust approximate degree.

Our main result (made precise in Theorem 4.2.1) is the following:

There exists a quantum algorithm that outputs x with high probability, using $O(n)$ invocations of the A_i algorithms (i.e., queries).

This result implies that every n -bit function f can be robustly quantum computed with $O(n)$ 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 results for OR and 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.

Our main 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 is to compute each instance with error probability reduced to $O(1/n)$, which costs another factor of $O(\log n)$. In fact, it follows from the $\Omega(n \log n)$ bound for PARITY that this factor of $n \log n$ is optimal when 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 [80, 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.

In addition to robust quantum algorithms, we also consider robustness for multivariate *polynomials* approximating Boolean functions. In general, there are many connections between the (quantum or classical) query complexity of an n -bit function and the degrees of n -variate polynomials that approximate it [38]. We consider two complementary definitions of robust polynomials. First, in analogy to the multiple-faulty-copies model, we can consider the usual approximating polynomial but on nm instead of just n binary variables, and require that if $\Pr[x_i = y_{i,j}] \geq 1 - \varepsilon$ for each $i = 1, \dots, n$, $j = 1, \dots, m$

then the polynomial p satisfies $\Pr[|p(y) - f(x)| \geq 1/3] \leq 1/3$. Secondly, we can define a robust polynomial for a Boolean function f to operate on n variables $z \in \mathbb{R}^n$, so that $|q(z_1, \dots, z_n) - f(x_1, \dots, x_n)| \leq 1/3$ whenever $x \in \{0, 1\}^n$ and $|z_i - x_i| \leq \varepsilon$ for all i . In Section 4.4 we show that the two types of robust polynomials are essentially equivalent, and that every non-robust approximating polynomial of degree d can be made robust at the cost of increasing its degree by a factor $O(\log d)$. Beals, Buhrman, Cleve, Mosca, and de Wolf [15] showed that every T -query quantum algorithm for f gives rise to a degree- $2T$ approximating polynomial for f , and similarly one can show that every T -query robust quantum algorithm for f induces a degree- $2T$ polynomial that approximates f robustly. This implies, for instance, that the robust degree of OR is $\Theta(\sqrt{n})$, and that every n -bit function has robust degree $O(n)$.

4.2 Robustly Recovering All n Bits

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

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

We assume 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 $|\alpha_i|^2 \leq \varepsilon$ if $x_i = 1$ and $|\alpha_i|^2 \geq 1 - \varepsilon$ if $x_i = 0$; $|\psi_i^0\rangle$ and $|\psi_i^1\rangle$ are arbitrary $(t-1)$ -qubit norm-1 quantum states. Every quantum algorithm can be expressed in this form by postponing measurements; every 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

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

which operates like A_i but outputs 1 when A_i would have output 0 and vice versa. Let

$$A_i(b) := \begin{cases} A_i & \text{if } b = 0 \\ \bar{A}_i & \text{if } b = 1 \end{cases}$$

Procedure RobustFind($n, \mathcal{A}, \varepsilon, \beta, \gamma, \delta$)

 $n \in \mathbb{N}, \mathcal{A} : n$ quantum algorithms, $\varepsilon, \beta, \gamma, \delta > 0$ **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 at least $1 - \delta$
2. if \mathcal{A} is ε -close to $x \in \{0, 1\}^n$ and if $i \neq \perp$, then $x_i = 1$ with probability at least $1 - \gamma$

Complexity:

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

If we plug the right bit x_i into A_i , then for all A_i we expect output 0: for the unique good $x \in \{0, 1\}^n$, $\mathcal{A}(x) := (A_1(x_1), \dots, A_n(x_n))$ is ε -close to 0^n by the following notion of closeness:

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

Our algorithm builds on a robust quantum search algorithm by Høyer, Mosca, and de Wolf [75]: the RobustFind subroutine above 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, our main program AllOutputs (as defined by the pseudo code on page 80) manages to construct the unique x with high probability. Note that RobustFind is the only quantum component of our otherwise classical algorithm.

Success probability The first step of our algorithm (Line 1 in AllOutputs) is to classically sample each i once and to store this initial approximation into a variable \tilde{x} . The following rounds of the algorithm refine \tilde{x} until with high probability it is correct (i.e., equal to x).

We call i a *bad* index if $i \in [n]$ and $\Pr[A_i \text{ outputs } x_i] \leq \varepsilon$. Let B_0 denote the random variable counting the number of bad indices after Line 1 in AllOutputs and let B_k denote the random variable of the number of bad

Procedure InitialGuess(n, \mathcal{A})

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

- 1: **for** $i \leftarrow$ to n **do**
 - 2: run A_i
 - 3: $\tilde{x}_i \leftarrow$ result of A_i
 - 4: **return** \tilde{x}
-

Procedure SampleBad($n, \mathcal{A}, \tilde{x}, r, \varepsilon, \beta, \gamma, \delta$)

 $n \in \mathbb{N}, \mathcal{A} : n$ algorithms, $\tilde{x} \in \{0, 1\}^n, r \in \mathbb{N}, \varepsilon, \beta, \gamma, \delta > 0$

- 1: **for** $\ell \leftarrow 1$ to r **do**
 - 2: $i \leftarrow$ RobustFind($n, \mathcal{A}(\tilde{x}), \varepsilon, \beta, \gamma, \delta$)
 - 3: **if** $i \neq \perp$ **then**
 - 4: $\tilde{x}_i \leftarrow 1 - \tilde{x}_i$
 - 5: **return** \tilde{x}
-

Procedure FindAllBad($n, \mathcal{A}, \tilde{x}, \varepsilon, \beta, \gamma, \delta$)

 $n \in \mathbb{N}, \mathcal{A} : n$ algorithms, $\tilde{x} \in \{0, 1\}^n, \varepsilon, \beta, \gamma, \delta > 0$

- 1: **repeat**
 - 2: $i \leftarrow$ RobustFind($n, \mathcal{A}(\tilde{x}), \varepsilon, \beta, \gamma, \delta$)
 - 3: **if** $i \neq \perp$ **then**
 - 4: $\tilde{x}_i \leftarrow 1 - \tilde{x}_i$
 - 5: **until** $i = \perp$
 - 6: **return** \tilde{x}
-

Procedure AllOutputs($n, \mathcal{A}, \varepsilon$)

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

- 1: $\tilde{x} \leftarrow$ InitialGuess(n, \mathcal{A})
 - 2: **for** $k \leftarrow 1$ to $\log(\varepsilon(\log n)^2)$ **do**
 - 3: $\varepsilon' \leftarrow \varepsilon/2^{k-1}$
 - 4: $\tilde{x} \leftarrow$ SampleBad($n, \mathcal{A}, \tilde{x}, 1.7\varepsilon'n, \varepsilon, 0.3\varepsilon', \frac{1}{8}, \frac{1}{8}$)
 - 5: $\tilde{x} \leftarrow$ FindAllBad($n, \mathcal{A}, \tilde{x}, \varepsilon, \frac{1}{10n}, \frac{1}{10n}, \frac{1}{10n}$)
 - 6: **return** \tilde{x}
-

indices after Line 4 in AllOutputs. By G_k we denote the event $B_k \leq n\varepsilon/2^{k-1}$. We have

$$\Pr[G_{k_{\max}}] \geq \Pr[G_0] \prod_{k=1}^{k_{\max}} \Pr[G_k|G_{k-1}] .$$

We now show that $\Pr[G_k|G_{k-1}]$ is large. For $k=0$, we know that $E[B_0] \leq \varepsilon n$ and $\Pr[B_0 \leq 2\varepsilon n] \geq 9/10$ by a Chernoff bound. In round k , we want to reduce the upper bound on the number of bad indices from $2n\varepsilon/2^{k-1}$ to $n\varepsilon/2^{k-1}$. If we have the maximum number of bad indices so that still G_k holds, we expect r repetitions of RobustFind to reduce the number of bad indices to

$$2 \frac{n\varepsilon}{2^{k-1}} - (1-\delta)((1-\gamma)r - \gamma r) \leq \frac{9}{10} \frac{n\varepsilon}{2^{k-1}}$$

therefore we choose

$$r := \frac{11}{10} \frac{1}{(1-\delta)(1-2\gamma)} \frac{n\varepsilon}{2^{k-1}} \approx 1.7 \frac{n\varepsilon}{2^{k-1}} .$$

On the other hand, if we have only a small number b of bad indices, it is likely that we will make many errors, so we would like

$$b + \gamma r \leq \frac{9}{10} \frac{n\varepsilon}{2^{k-1}} .$$

This is satisfied by choosing $b := 0.3n\varepsilon/2^{k-1}$; this choice of b also ensures that we never get as few as b bad indices if we start the round with $2n\varepsilon/2^{k-1}$ bad indices.

We tune RobustFind to find bad indices with probabilities δ and γ if there are at least b bad indices. Hence, in the extreme cases of either having exactly $2n\varepsilon/2^{k-1}$ or less than b bad indices, we expect to arrive at at most $(9/10) \cdot n\varepsilon/2^{k-1}$ bad indices, and this holds for the intermediary cases as well. By a Chernoff-type argument, the probability that we are a constant factor $10/9$ away from the expectation is exponentially small in the number r of samples, therefore, with $k_{\max} = \log(\varepsilon(\log n)^2)$, we have

$$\Pr[G_k|G_{k-1}] \geq 1 - e^{-\Omega(n/\log n)}$$

and

$$\begin{aligned} \Pr[G_{k_{\max}}] &\geq \Pr[G_0] \left(1 - e^{-\Omega(n/\log n)}\right)^{k_{\max}} \\ &\geq \frac{9}{10} \left(1 - \frac{k_{\max}}{e^{\Omega(n/\log n)}}\right) = \frac{9}{10} - o(1) . \end{aligned}$$

Hence, for large n with probability $8/10$ we have at most $n/(\log n)^2$ bad indices at Line 6 in AllOutputs. 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 $\geq (8/10) \cdot (9/10) > 2/3$.

Complexity We bound the number of queries to f in SampleBad as follows:

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

for some constants C, C' . The call to FindAllBad results in

$$O\left(\frac{1}{(\frac{1}{2} - \epsilon)^2} \sqrt{(\log n)^2} \log n \cdot \left(\frac{n}{(\log n)^2}\right)\right) = O\left(\frac{n}{(\frac{1}{2} - \epsilon)^2}\right)$$

many queries. Therefore, the total query complexity of AllOutputs also is $O(n/(1/2 - \epsilon)^2)$.

Consequences Once we have recovered the input x , we can compute an arbitrary function of x without further queries.

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

In particular, PARITY can be robustly quantum computed with $O(n)$ queries while it takes $\Omega(n \log n)$ queries classically [53].

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.

4.2.4. COROLLARY (DIRECT SUM). *If there exists a T -query bounded-error quantum algorithm for f , then there is an $O(Tn)$ -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.

Finally, 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 $\Gamma(f) := \min\{|2k - n + 1| : f \text{ flips value if the Hamming weight of the input changes from } k \text{ to } k + 1\}$. The non-robust algorithm for computing f with $O(\sqrt{n(n - \Gamma(f))})$ queries [15, Theorem 4.10] can be made robust by a similar algorithm as the one used in the proof of Theorem 4.2.1, giving:

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

4.3 The Multiple-Faulty-Copies Model

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 [115] where we have a sequence $y_{i,1}, \dots, y_{i,m}$ of faulty 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 $m := O((\log n)/\varepsilon^2)$ implies that the average $\bar{y}_i := \sum_{j=1}^m y_{i,j}/m$ is close to x_i with very high probability:

$$\Pr[|\bar{y}_i - x_i| \geq 2\varepsilon] \leq \frac{1}{100n} .$$

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 to the $y_{i,j}$:

$$A_i : |0^{\log(m)+1}\rangle \mapsto \frac{1}{\sqrt{m}} \sum_{j=1}^m |j\rangle|y_{i,j}\rangle .$$

Measuring the last qubit of the resulting state gives x_i with probability at least $1 - 2\varepsilon$. Hence, we can run our algorithm from Section 4.2 and recover x using $O(n)$ queries to the $y_{i,j}$. Similarly, all consequences mentioned in the last section hold for this multiple-faulty-copies model as well.

4.4 Robust Polynomials

In this section we study robust polynomials, of two different but essentially equivalent types. The first type follows the many-faulty-copies model.

4.4.1. DEFINITION. 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 $\Pr[y_{i,j} = x_i] \geq 1 - \varepsilon$ for each $1 \leq j \leq m$.

4.4.2. DEFINITION. A *type-1* (ε, m) -robust polynomial for the Boolean function $f(x_1, \dots, x_n)$ 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 $-1/3 \leq p(v) \leq 4/3$.

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 any other constant.

4.4.3. LEMMA. *If there is a type-1 (ε, m) -robust polynomial of degree d for f , then for some $m' = O(m)$ there exists a type-1 (ε, m') -robust polynomial p of degree $O(d)$ 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 every $v \in \{0, 1\}^{nm'}$, $-1/9 \leq p(v) \leq 10/9$.*

Proof. Let p_0 denote the type-1 (ε, m) -robust polynomial that we start with. The single-variate polynomial $g(a) := (2a-1)(1+a(2+(a/22)(1+45a(a-2))))$ has the property that $-1/9 \leq g^3(a) \leq 1/9$ for $-1/3 \leq a \leq 1/3$ and $8/9 \leq g^3(a) \leq 10/9$ for $2/3 \leq a \leq 4/3$; here $g^t(a)$ denotes the t -fold application of g .

$$g^t(a) := \underbrace{g(g(\cdots g(a)))}_t .$$

Therefore $p_1(y) := g^3(p_0(y))$ satisfies $|p_1(y) - f(x)| \leq 1/9$ whenever $|p_0(y) - f(x)| \leq 1/3$.

For some r to be determined later and an arbitrary $x \in \{0, 1\}^n$, we use r independent (ε, m) perturbations y_k of x , $1 \leq k \leq r$. Let B denote the random variable counting the number of indices k so that $|p_1(y_k) - f(x)| \geq 1/9$. Choosing r sufficiently large, a Chernoff bound implies $\Pr[B \geq 13r/36] \leq 1/9$. Therefore

$$\begin{aligned} \Pr \left[\left| \frac{1}{r} \sum_{k=1}^r p_1(y_k) - f(x) \right| \geq \frac{17}{36} \right] &\leq \Pr \left[\sum_{k=1}^r |p_1(y_k) - f(x)| \geq \frac{17}{36} r \right] \\ &\leq \Pr \left[\frac{10}{9} B + (r - B) \frac{1}{9} \geq \frac{17}{36} r \right] \\ &= \Pr \left[B \geq \frac{13}{36} r \right] \leq \frac{1}{9} \end{aligned}$$

Let $p_2(y_1, \dots, y_r) := \frac{1}{r} \sum_{k=1}^r p_1(y_k)$. We move closer to f in the “good” case: $p(y_1, \dots, y_r) := g^6(p_2(y_1, \dots, y_r))$ satisfies

$$\Pr \left[|p(y_1, \dots, y_r) - f(x)| \geq \frac{1}{9} \right] \leq \frac{1}{9} \quad \text{and} \quad -\frac{1}{9} \leq p(v) \leq \frac{10}{9}$$

for all $v \in \{0, 1\}^{nm}$. Now we are done: with $m' := rm$ we have that y_1, \dots, y_r is an (ε, m') perturbation of x and $\deg(p) = O(\deg(p_0))$. \square

The second kind of robust polynomial is the following:

4.4.4. DEFINITION. A *type-2 ε -robust polynomial* for the Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is a real polynomial q in n variables $z_1, \dots, z_n \in \mathbb{R}$ so that for every $x \in \{0, 1\}^n$ and $z \in \mathbb{R}^n$ we have $|q(z) - f(x)| \leq 1/3$ if $|z_i - x_i| \leq \varepsilon$ for all $i \in [n]$. If $\varepsilon = 0$, then q is called an *approximating polynomial* for f .

4.4.5. THEOREM. For every type-2 ε -robust polynomial of degree d for f there is a type-1 $(\varepsilon/2, O(\log(n)/(1/2 - \varepsilon)^2))$ -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 $O(d)$ for f .

Proof. Let p be a type-2 ε -robust polynomial of degree d for f . As in Section 4.3, we choose $m = O(\log(n)/(1/2 - \varepsilon)^2)$. If each $y_{i,j}$ is wrong with probability $\leq \varepsilon/2$, then with probability at least $2/3$, the averages \bar{y}_i will satisfy $|\bar{y}_i - x_i| \leq \varepsilon$ for all $i \in [n]$. Hence the polynomial $p(\bar{y}_1, \dots, \bar{y}_n)$ will be a type 1 $(\varepsilon/2, O(\log(n)/(1/2 - \varepsilon)^2))$ -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 4.4.3, we boost the approximation parameters to obtain a type-1 (ε, m') -robust polynomial p of degree $O(d)$, with $m' = O(m)$, so that for every $x \in \{0, 1\}^n$ 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 $q(z) := E[p(y)]$. This q is a polynomial in z , because $E[p(y)] = p(E[y])$ and $E[y_{i,j}] = z_i$. Moreover, if for z there exists $x \in \{0, 1\}^n$ with $|z_i - x_i| \leq \varepsilon$ 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

$$\begin{aligned} |f(x) - q(z)| &= \left| \sum_{v \in \{0,1\}^{nm}} \Pr[y = v] (f(x) - p(v)) \right| \\ &\leq \left| \sum_{v \in V} \Pr[y = v] (f(x) - p(v)) \right| + \left| \sum_{v \notin V} \Pr[y = v] \left(1 + \frac{1}{9}\right) \right| \\ &\leq \frac{8}{9} \cdot \frac{1}{9} + \frac{1}{9} \cdot \frac{10}{9} < \frac{1}{3}. \end{aligned}$$

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

4.4.6. DEFINITION. For $f : \{0, 1\}^n \rightarrow \{0, 1\}$, let $\text{rdeg}_1(f)$ denote the minimum degree of the type-1 $(1/3, 5 \log n)$ polynomials for f , $\text{rdeg}_2(f)$ be the minimum degree of the type-2 $1/3$ -robust polynomials approximating f , and $\widetilde{\text{deg}}(f)$ be the minimum degree among all approximating polynomials for f .

Note that in Definition 4.4.2 we require for type-1 polynomials p that for each Boolean assignment $v \in \{0, 1\}^{nm}$ to the (possibly real) variables, the polynomial value $p(v)$ between $-1/3$ and $4/3$. Because of this totality requirement, the following corollaries are given for total Boolean functions.

4.4.7. COROLLARY. $\text{rdeg}_1(f) = \Theta(\text{rdeg}_2(f))$ for every (total) Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$.

4.4.8. COROLLARY. $\text{rdeg}_{1,2}(f) = O(\widetilde{\text{deg}}(f) \log n)$ for every (total) Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$.

Using the notion of *certificate complexity* $C(f)$ and its polynomial relation to $\widetilde{\text{deg}}(f)$, one can strengthen Corollary 4.4.8 to the following theorem [36].

4.4.9. THEOREM. $\text{rdeg}_{1,2}(f) = O(\widetilde{\text{deg}}(f) \cdot \log \widetilde{\text{deg}}(f))$.

4.5 Discussion and Open Problems

In contrast to the classical case, we do not know of any function where making a quantum algorithm or polynomial robust costs more than a constant factor. In the case of symmetric functions, such a constant overhead suffices. 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 have chosen our model of a noisy query so that we can coherently make a query and reverse it. An open question is whether the advantage of quantum algorithms can be maintained for “decohering” queries, like the first model proposed in the introduction. It is not clear to what extent non-robust quantum algorithms can be made resilient against such random noise, 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.