INSTITUTE OF PHYSICS PUBLISHING JOURNAL OF PHYSICS A: MATHEMATICAL AND GENERAL

J. Phys. A: Math. Gen. 34 (2001) 6741–6754 PII: S0305-4470(01)19314-5

Average-case quantum query complexity*

Andris Ambainis[1] **and Ronald de Wolf**[2]

[1] Computer Science Department, University of California, Berkeley, CA 94720, USA
[2] CWI, PO Box 94079, 1090 GB Amsterdam, The Netherlands

E-mail: ambainis@cs.berkeley.edu and rdewolf@cwi.nl

Received 21 November 2000
Published 24 August 2001
Online at stacks.iop.org/JPhysA/34/6741

Abstract

We compare classical and quantum query complexities of total Boolean functions. It is known that for *worst-case* complexity, the gap between quantum and classical can be at most polynomial. We show that for *average-case* complexity under the uniform distribution, quantum algorithms can be exponentially faster than classical algorithms. Under non-uniform distributions the gap can even be super-exponential. We also prove some general bounds for average-case complexity and show that the average-case quantum complexity of MAJORITY under the uniform distribution is nearly quadratically better than the classical complexity.

PACS numbers: 03.67.-a, 02.10.Ab, 02.10.De, 02.50.Cw, 03.65.Ta

1. Introduction

The field of quantum computation studies the power of computers based on quantum mechanical principles. So far, most quantum algorithms—and *all* physically implemented ones—have operated in the so-called *black-box* setting. In the black-box model, the input of the function f that we want to compute can only be accessed by means of queries to a 'black box'. This returns the ith bit of the input when queried on i. The complexity of computing f is measured by the required number of queries. In this setting we want quantum algorithms that use significantly fewer queries than the best classical algorithms. Examples of quantum black-box algorithms that are provably better than any classical algorithm can be found in [6, 7, 9, 12, 14, 25]. Even Shor's quantum algorithm for period-finding, which is the core of his efficient factoring algorithm [24], can be viewed as a black-box algorithm [11].

We restrict our attention to computing total Boolean functions f on N variables. The query complexity of f depends on the kind of error one allows. For example, we can distinguish between exact computation, zero-error computation (a.k.a. Las Vegas) and

* A preliminary version of this paper appeared in the *Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science (STACS'2000) (Lecture Notes in Computer Science* vol 1770) (Berlin: Springer).

0305-4470/01/356741+14$30.00

bounded-error computation (Monte Carlo). In each of these models, *worst-case* complexity is usually considered: the complexity is the number of queries required for the 'hardest' input. Let $D(f)$, $R(f)$ and $Q(f)$ denote the worst-case query complexity of computing f for classical deterministic algorithms, classical randomized bounded-error algorithms and quantum bounded-error algorithms, respectively. More precise definitions will be given in the next section. Since quantum bounded-error algorithms are at least as powerful as classical bounded-error algorithms, and classical bounded-error algorithms are at least as powerful as deterministic algorithms, we have $Q(f) \leqslant R(f) \leqslant D(f)$. The main quantum success here is Grover's algorithm [14]. It can compute the OR-function with bounded error using $\Theta(\sqrt{N})$ queries (which is optimal [4, 5, 27]). Thus $Q(\mathrm{OR}) \in \Theta(\sqrt{N})$, whereas $D(\mathrm{OR}) = N$ and $R(\mathrm{OR}) \in \Theta(N)$. This is the biggest gap known between quantum and classical worst-case complexities for total functions. (In contrast, for *partial* Boolean functions the gap can be much bigger [11, 12, 25].) In fact, it is known that the gap between $D(f)$ and $Q(f)$ is at most polynomial for *every* total f: $D(f) \in \mathrm{O}(Q(f)^6)$ [3]. This is similar to the best known relation between classical deterministic and randomized algorithms: $D(f) \in \mathrm{O}(R(f)^3)$ [21].

Given some probability distribution μ on the set of inputs $\{0, 1\}^N$ one may also consider *average-case* complexity instead of worst-case complexity. Average-case complexity concerns the *expected* number of queries needed when the input is distributed according to μ. If the hard inputs receive little μ-probability, then average-case complexity can be significantly smaller than worst-case complexity. Let $D^\mu(f)$, $R^\mu(f)$ and $Q^\mu(f)$ denote the average-case analogues of $D(f)$, $R(f)$ and $Q(f)$, respectively, to be defined more precisely in the next section. Again $Q^\mu(f) \leqslant R^\mu(f) \leqslant D^\mu(f)$. The objective of this paper is to compare these measures and to investigate the possible gaps between them. Our main results are:

- Under uniform μ, $Q^\mu(f)$ and $R^\mu(f)$ can be super-exponentially smaller than $D^\mu(f)$.
- Under uniform μ, $Q^\mu(f)$ can be exponentially smaller than $R^\mu(f)$. Thus the polynomial relation that holds between quantum and classical query complexities in the case of worst-case complexity [3] does not carry over to the average-case setting.
- Under non-uniform μ the gap can be even larger: we give distributions μ where $Q^\mu(\mathrm{OR})$ is constant, whereas $R^\mu(\mathrm{OR})$ is almost $\sqrt{N}$.
- For every f and μ, $R^\mu(f)$ is lower bounded by the expected *block sensitivity* $E_\mu[bs(f)]$ and $Q^\mu(f)$ is lower bounded by $E_\mu[\sqrt{bs(f)}]$.
- For the MAJORITY function under uniform μ, we have that $Q^\mu(f) \in \mathrm{O}(\sqrt{N}(\log N)^2)$ and $Q^\mu(f) \in \Omega(\sqrt{N})$. In contrast, $R^\mu(f) \in \Omega(N)$.
- For the PARITY function, the gap between Q^μ and R^μ can be quadratic, but not more. Under uniform μ, PARITY has $Q^\mu(f) \in \Omega(N)$.

2. Definitions

Let $f : \{0, 1\}^N \to \{0, 1\}$ be a Boolean function. This function is *symmetric* if $f(X)$ only depends on $|X|$, the Hamming weight (the number of ones) of X. We will in particular consider the following symmetric functions: $\mathrm{OR}(X) = 1$ iff $|X| \geqslant 1$; $\mathrm{MAJ}(X) = 1$ iff $|X| > N/2$; $\mathrm{PARITY}(X) = 1$ iff $|X|$ is odd. If $X \in \{0, 1\}^N$ is an input and S a set of (indices of) variables, we use X^S to denote the input obtained by flipping the values of the S-variables in X. The *block sensitivity* $bs_X(f)$ of f on an input X is the maximal number b for which there are b disjoint sets of variables $S_1, \ldots, S_b$ such that $f(X) \neq f(X^{S_i})$ for all $1 \leqslant i \leqslant b$. The block sensitivity $bs(f)$ of f is $\max_X bs_X(f)$.

We are interested in the question of how many bits of the input have to be queried in order to compute f, either for the worst- or average-case input. We assume familiarity with

classical computation and briefly sketch the definition of quantum query algorithms. For a general introduction to quantum computing, see the book of Nielsen and Chuang [20]. For more details about (quantum) query complexity we refer to [10].

An m-qubit state is a 2^m-dimensional unit vector of complex numbers, written $\sum_{x\in\{0,1\}^m} \alpha_x|x\rangle$. The complex number α_x is called the *amplitude* of the basis state $|x\rangle$. A T-query quantum algorithm corresponds to a unitary transformation

$$A = U_T \mathrm{O} U_{T-1} \mathrm{O} \cdots U_1 \mathrm{O} U_0.$$

Here the U_j are unitary transformations on m qubits. These U_j are independent of the input. Each O corresponds to a query to the input $X \in \{0,1\}^N$, formalized as the unitary transformation

$$|i, b, z\rangle \to |i, b \oplus x_i, z\rangle.$$

Here $i \in \{1, \ldots, N\}$, $b \in \{0,1\}$, $\oplus$ is addition modulo 2 and $z \in \{0,1\}^{m-\log N-1}$ is the workspace, which remains unaffected by the query. Intuitively, O just gives us the bit x_i when queried on i. We will sometimes use the word 'oracle' to refer to X as well as to the corresponding O. The initial state of the algorithm is the all-zero state $|0^m\rangle$. The final state is $A|0^m\rangle$, which depends on the input X via the T queries that are made. A measurement of a dedicated *output bit* of the final state will yield the output. It can be shown that this linear-algebraic quantum model is at least as strong as classical randomized computation: any classical T-query randomized algorithm can be simulated by a T-query quantum algorithm having the same error probabilities.

As described above, the quantum algorithm will make exactly T queries on *every* input X. Since we are interested in an average-case number of queries and the required number of queries will depend on the input X, we need to allow the algorithm to give an output after fewer than T queries. We will do that by measuring, after each U_j, a dedicated *flag-qubit* of the intermediate state at that point (this measurement may alter the state). This bit indicates whether the algorithm is already prepared to stop and output a value. If this bit is 1, then we measure the output bit, output its value $A(X) \in \{0,1\}$ and stop; if the flag-bit is 0 we let the algorithm continue with the next query O and U_{j+1}. Note that the number of queries that the algorithm makes on input X is now a random variable, since it depends on the probabilistic outcome of measuring the flag-qubit after each step. We use $T_A(X)$ to denote the *expected* number of queries that A makes on input X. The Boolean output $A(X)$ of the algorithm is a random variable as well.

We mainly focus on three kinds of algorithm for computing f: classical *deterministic*, classical *randomized* bounded-error and *quantum* bounded-error algorithms. Let $\mathcal{D}(f)$ denote the set of classical *deterministic* algorithms that compute f. Let $\mathcal{R}(f) = \{\text{classical } A \mid \forall X \in \{0,1\}^N : \Pr[A(X) = f(X)] \geqslant 2/3\}$ be the set of classical *randomized* algorithms that compute f with bounded error probability. The error probability one-third is not essential; it can be reduced to any small ε by running the algorithm $O(\log(1/\varepsilon))$ times and outputting the majority answer of those runs. Similarly we let $\mathcal{Q}(f) = \{\text{quantum } A \mid \forall X \in \{0,1\}^N : \Pr[A(X) = f(X)] \geqslant 2/3\}$ be the set of bounded-error *quantum* algorithms for f. We define the following worst-case complexities:

$$D(f) = \min_{A\in\mathcal{D}(f)} \max_{X\in\{0,1\}^N} T_A(X)$$

$$R(f) = \min_{A\in\mathcal{R}(f)} \max_{X\in\{0,1\}^N} T_A(X)$$

$$Q(f) = \min_{A\in\mathcal{Q}(f)} \max_{X\in\{0,1\}^N} T_A(X).$$

$D(f)$ is also known as the *decision tree complexity* of f and $R(f)$ as the *bounded-error* decision tree complexity of f. Since quantum computation generalizes randomized

computation and randomized computation generalizes deterministic computation, we have $Q(f) \leqslant R(f) \leqslant D(f) \leqslant N$ for all f. The three worst-case complexities are polynomially related: $D(f) \in O(R(f)^3)$ [21] and $D(f) \in O(Q(f)^6)$ [3] for all total f.

Let $\mu : \{0,1\}^N \to [0,1]$ be a probability distribution. We define the *average-case complexity* of an algorithm A with respect to a distribution μ as

$$T_A^\mu = \sum_{X \in \{0,1\}^N} \mu(X) T_A(X).$$

The average-case deterministic, randomized and quantum complexities of f with respect to μ are

$$D^\mu(f) = \min_{A \in \mathcal{D}(f)} T_A^\mu$$

$$R^\mu(f) = \min_{A \in \mathcal{R}(f)} T_A^\mu$$

$$Q^\mu(f) = \min_{A \in \mathcal{Q}(f)} T_A^\mu.$$

Note that the algorithms still have to satisfy the appropriate output requirements (such as outputting $f(X)$ with probability $\geqslant 2/3$ in the case of R^μ or Q^μ) on *all* inputs X, even on X that have $\mu(X) = 0$. Clearly $Q^\mu(f) \leqslant R^\mu(f) \leqslant D^\mu(f) \leqslant N$ for all μ and f. Our goal is to examine how large the gaps between these measures can be, in particular for the uniform distribution $\mathrm{unif}(X) = 2^{-N}$.

The above treatment of average-case complexity is the standard one used in average-case analysis of algorithms [26]. One counter-intuitive consequence of these definitions, however, is that the average-case performance of polynomially related algorithms can be superpolynomially apart (we will see this happen in section 5). This seemingly paradoxical effect makes these definitions unsuitable for dealing with polynomial-time reducibilities and average-case complexity classes, which is what led Levin to his alternative definition of 'polynomial time on average' [16][3]. Nevertheless, we feel our definitions are the appropriate ones for our query complexity setting: they *are* just the average numbers of queries that one needs when the input is drawn according to distribution μ.

3. Super-exponential gap between $D^{\mathrm{unif}}(f)$ and $Q^{\mathrm{unif}}(f)$

Before comparing the power of classical and quantum computing, we first compare the power of *deterministic* and *bounded-error* algorithms. It is not hard to show that $D^{\mathrm{unif}}(f)$ can be much larger then $R^{\mathrm{unif}}(f)$ and $Q^{\mathrm{unif}}(f)$:

Theorem 3.1. *Define f on N variables such that $f(X) = 1$ iff $|X| \geqslant N/10$. Then $Q^{\mathrm{unif}}(f)$ and $R^{\mathrm{unif}}(f)$ are $O(1)$ and $D^{\mathrm{unif}}(f) \in \Omega(N)$.*

Proof. Suppose we randomly sample k bits of the input. Let $a = |X|/N$ denote the fraction of ones in the input and $\tilde{a}$ the fraction of ones in the sample. The Chernoff bound (see e.g. [1]) implies that there is a constant $c > 0$ such that

$$\Pr[\tilde{a} \langle 2/10 | a \geqslant 3/10] \leqslant 2^{-ck}.$$

Now consider the following randomized algorithm for f:

(1) Let $i = 100$.
(2) Sample $k_i = i/c$ bits. If the fraction $\tilde{a}_i$ of ones is $\geqslant 2/10$, then output 1 and stop.
(3) If $i < \log N$, then increase i by 1 and repeat step 2.

[3] We thank Umesh Vazirani for drawing our attention to this.

(4) If $i \geqslant \log N$, then count $|X|$ exactly using N queries and output the correct answer.

It is easy to see that this is a bounded-error algorithm for f. Let us bound its average-case complexity under the uniform distribution.

If $a \geqslant 3/10$, the expected number of queries for step 2 is

$$\sum_{i=100}^{\log N} \Pr[\tilde{a}_1 \leqslant 2/10, \ldots, \tilde{a}_{i-1} \leqslant 2/10 \mid a \geqslant 3/10] \cdot \frac{i}{c}$$
$$\leqslant \sum_{i=100}^{\log N} \Pr[\tilde{a}_{i-1} \leqslant 2/10 \mid a \geqslant 3/10] \cdot \frac{i}{c}$$
$$\leqslant \sum_{i=100}^{\log N} 2^{-(i-1)} \cdot \frac{i}{c} \in O(1).$$

The probability that step 4 is needed (given $a \geqslant 3/10$) is at most $2^{-c \log N/c} = 1/N$. This adds $\frac{1}{N}N = 1$ to the expected number of queries.

Under the uniform distribution, the probability of the event $a < 3/10$ is at most $2^{-c'N}$ for some constant c'. This case contributes at most $2^{-c'N}(N + (\log N)^2) \in o(1)$ to the expected number of queries. Thus in total the algorithm uses $O(1)$ queries on average, hence $R^{\text{unif}}(f) \in O(1)$. Since $Q^{\text{unif}}(f) \leqslant R^{\text{unif}}(f)$, we also have $Q^{\text{unif}}(f) \in O(1)$.

Since a deterministic classical algorithm for f must be correct on every input X, it is easy to see that it must make at least $N/10$ queries on every input, hence $D^{\text{unif}}(f) \geqslant N/10$. □

Accordingly, we can have huge gaps between $D^{\text{unif}}(f)$ and $Q^{\text{unif}}(f)$. However, this example tells us nothing about the gaps between quantum and classical bounded-error algorithms. In the next section we exhibit an f where $Q^{\text{unif}}(f)$ is exponentially smaller than the classical bounded-error complexity $R^{\text{unif}}(f)$.

4. Exponential gap between $R^{\text{unif}}(f)$ and $Q^{\text{unif}}(f)$

4.1. The function

We use the following modification of Simon's problem [25][4]:

Input: $X = (x_1, \ldots, x_{2^n})$, where each $x_i \in \{0, 1\}^n$.
Output: $f(X) = 1$ iff there is a non-zero $k \in \{0, 1\}^n$ such that for all $i \in \{0, 1\}^n$ we have $x_{i \oplus k} = x_i$.

Here we treat $i \in \{0, 1\}^n$ both as an n-bit string and as a number between 1 and 2^n, and $\oplus$ denotes bitwise XOR. Note that this function is total (unlike Simon's). Formally, f is not a Boolean function because the variables are $\{0, 1\}^n$-valued. However, we can replace every variable x_i by n Boolean variables and then f becomes a Boolean function of $N = n2^n$ variables. The number of queries needed to compute the Boolean function is at least the number of queries needed to compute the function with $\{0, 1\}^n$-valued variables (because we can simulate a query to the Boolean oracle by means of a query to the $\{0, 1\}^n$-valued input-variables, just ignoring the $n - 1$ bits that we are not interested in) and at most n times the number of queries to the $\{0, 1\}^n$-valued oracle (because one $\{0, 1\}^n$-valued query can be simulated using n Boolean queries). As the numbers of queries are so closely related, it does not make a big difference whether we use the $\{0, 1\}^n$-valued oracle or the Boolean oracle. For simplicity we count queries to the $\{0, 1\}^n$-valued oracle.

[4] The preprint [15] independently proves a related but incomparable result about another Simon modification.

We are interested in the average-case complexity of this function. The main result is the following exponential gap, to be proven in the next sections:

Theorem 4.1. *For f as above, $Q^{\text{unif}}(f) \leqslant 22n + 1$ and $R^{\text{unif}}(f) \in \Omega(2^{n/2})$.*

4.2. Quantum upper bound

The quantum algorithm is similar to Simon's. Start with the 2-register superposition $\sum_{i\in\{0,1\}^n} |i\rangle|0\rangle$ (for convenience we ignore normalizing factors). Apply the oracle once to obtain

$$\sum_{i\in\{0,1\}^n} |i\rangle|x_i\rangle.$$

Measuring the second register gives some j and collapses the first register to

$$\sum_{i:x_i=j} |i\rangle.$$

A *Hadamard transform* H maps bits $|b\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + (-1)^b|1\rangle)$. Applying this to each qubit of the first register gives

$$\sum_{i:x_i=j} \sum_{i'\in\{0,1\}^n} (-1)^{(i,i')}|i'\rangle. \tag{1}$$

Here (a, b) denotes inner product mod 2; if $(a, b) = 0$ we say a and b are orthogonal.

If $f(X) = 1$, then there is a non-zero k such that $x_i = x_{i\oplus k}$ for all i. In particular, $x_i = j$ iff $x_{i\oplus k} = j$. Then the final state (1) can be rewritten as

$$\sum_{i'\in\{0,1\}^n} \sum_{i:x_i=j} (-1)^{(i,i')}|i'\rangle = \sum_{i'\in\{0,1\}^n} \left(\sum_{i:x_i=j} \tfrac{1}{2}((-1)^{(i,i')} + (-1)^{(i\oplus k,i')}) \right)|i'\rangle$$

$$= \sum_{i'\in\{0,1\}^n} \left(\sum_{i:x_i=j} \frac{(-1)^{(i,i')}}{2}(1 + (-1)^{(k,i')}) \right)|i'\rangle.$$

Notice that $|i'\rangle$ has non-zero amplitude only if $(k, i') = 0$. Hence if $f(X) = 1$, then measuring the final state gives some i' orthogonal to the unknown k.

To decide whether $f(X) = 1$, we repeat the above process $m = 22n$ times. Let $i_1, \ldots, i_m \in \{0, 1\}^n$ be the results of the m measurements. If $f(X) = 1$, there must be a non-zero k that is orthogonal to all i_r. Compute the subspace $S \subseteq \{0, 1\}^n$ that is generated by $i_1, \ldots, i_m$ (i.e. S is the set of binary vectors obtained by taking linear combinations of $i_1, \ldots, i_m$ over $GF(2)$). If $S = \{0, 1\}^n$, then the only k that is orthogonal to all i_r is $k = 0^n$, so then we know that $f(X) = 0$. If $S \neq \{0, 1\}^n$, we just query all 2^n values $x_{0\ldots0}, \ldots, x_{1\ldots1}$ and then compute $f(X)$. Of course, this latter step is very expensive, but it is needed only rarely:

Lemma 4.2. *Assume that $X = (x_{0\ldots0}, \ldots, x_{1\ldots1})$ is chosen uniformly at random from $\{0, 1\}^N$. Then, with probability at least $1 - 2^{-n}$, $f(X) = 0$ and the measured $i_1, \ldots, i_m$ generate $\{0, 1\}^n$.*

Proof. It can be shown by a small modification of [1, theorem 5.1, p 91] that with probability at least $1 - 2^{-c2^n}$ ($c > 0$), there are at least $2^n/8$ values j such that $x_i = j$ for exactly one $i \in \{0, 1\}^n$ (and hence $f(X) = 0$). We assume that this is the case in the following.

If $i_1, \ldots, i_m$ generate a proper subspace of $\{0, 1\}^n$, then there is a non-zero $k \in \{0, 1\}^n$ that is orthogonal to this subspace. We estimate the probability that this happens. Consider some fixed non-zero vector $k \in \{0, 1\}^n$. The probability that i_1 and k are orthogonal is at most $\frac{15}{16}$, as follows. With probability at least $1/8$, the measurement of the second register gives j such

that $f(i) = j$ for a unique i. In this case, the measurement of the final superposition (1) gives a uniformly random i'. The probability that a uniformly random i' has $(k, i') \neq 0$ is one-half. Therefore, the probability that $(k, i_l) = 0$ is at most $1 - \frac{1}{8} \cdot \frac{1}{2} = \frac{15}{16}$.

The vectors $i_1, \ldots, i_m$ are chosen independently. Therefore, the probability that k is orthogonal to each of them is at most $(\frac{15}{16})^m = (\frac{15}{16})^{22n} < 2^{-2n}$. There are $2^n - 1$ possible non-zero k, so the probability that there is a k which is orthogonal to each of $i_1, \ldots, i_m$ is $\leqslant (2^n - 1)2^{-2n} < 2^{-n}$. □

Note that this algorithm is actually a *zero-error* algorithm: it always outputs the correct answer. Its expected number of queries on a uniformly random input is at most $m = 22n$ for generating $i_1, \ldots, i_m$ and at most $\frac{1}{2^n} 2^n = 1$ for querying all the x_i if the first step does not give $i_1, \ldots, i_m$ that generate $\{0, 1\}^n$. This completes the proof of the first part of theorem 4.1. In contrast, in the appendix we show that the *worst-case* zero-error quantum complexity of f is $\Omega(N)$, which is near maximal.

4.3. Classical lower bound

Let D_1 be the uniform distribution over all inputs $X \in \{0, 1\}^N$ and D_2 be the uniform distribution over all X for which there is a unique $k \neq 0$ such that $x_i = x_{i \oplus k}$ (and hence $f(X) = 1$). We say an algorithm A *distinguishes* between D_1 and D_2 if the average probability that A outputs 0 is $\geqslant 2/3$ under D_1 and the average probability that A outputs 1 is $\geqslant 2/3$ under D_2.

Lemma 4.3. *If there is a bounded-error algorithm A that computes f with $m = T_A^{\text{unif}}$ queries on average, then there is an algorithm that distinguishes between D_1 and D_2 and uses $O(m)$ queries on all inputs.*

Proof. Without loss of generality we assume A has error probability $\leqslant 1/10$. To distinguish D_1 and D_2, we run A until it stops or makes $10m$ queries. If it stops, we output the result of A. If it makes $10m$ queries and has not stopped yet, we output 1.

Under D_1, the probability that A outputs 1 is at most $1/10 + o(1)$ ($1/10$ is the maximum probability of error on an input with $f(X) = 0$ and $o(1)$ is the probability of getting an input with $f(X) = 1$), so the probability that A outputs 0 is at least $9/10 - o(1)$. The average probability (under D_1) that A does not stop before $10m$ queries is at most one-tenth, for otherwise the average number of queries would be more than $\frac{1}{10}(10m) = m$. Therefore the probability under D_1 that A outputs 0 after at most $10m$ queries is at least $(9/10 - o(1)) - 1/10 = 4/5 - o(1)$. In contrast, the D_2-probability that A outputs 0 is $\leqslant 1/10$ because $f(X) = 1$ for any input X from D_2. This shows that we can distinguish D_1 from D_2. □

Lemma 4.4. *A classical randomized algorithm A that makes $m \in o(2^{n/2})$ queries cannot distinguish between D_1 and D_2.*

Proof. For a random input from D_1, the probability that all answers to m queries are different is

$$1 \cdot \left(1 - \frac{1}{2^n}\right) \cdots \left(1 - \frac{(m-1)}{2^n}\right) \geqslant 1 - \sum_{i=1}^{m-1} \frac{i}{2^n} = 1 - \frac{m(m-1)}{2^{n+1}} = 1 - o(1).$$

For a random input from D_2, the probability that there is an i such that A queries both x_i and $x_{i \oplus k}$ (k is the hidden vector) is $\leqslant \binom{m}{2}/(2^n - 1) \in o(1)$, since:

(1) for every pair of distinct i, j, the probability that $i = j \oplus k$ is $1/(2^n - 1)$ and

(2) since A queries only m of the x_i, it queries only $\binom{m}{2}$ distinct pairs i, j.

If no pair $x_i, x_{i\oplus k}$ is queried, the probability that all answers are different is

$$1 \cdot \left(1 - \frac{1}{2^{n-1}}\right) \cdots \left(1 - \frac{(m-1)}{2^{n-1}}\right) = 1 - o(1).$$

It is easy to see that all sequences of m different answers are equally likely. Therefore, for both distributions D_1 and D_2, we get a uniformly random sequence of m different values with probability $1 - o(1)$ and something else with probability $o(1)$. Thus A cannot 'see' the difference between D_1 and D_2 with sufficient probability to distinguish between them. □

The second part of theorem 4.1 now follows: a classical algorithm that computes f with an average number of m queries can be used to distinguish between D_1 and D_2 with $O(m)$ queries (lemma 4.3), but then $O(m) \in \Omega(2^{n/2})$ (lemma 4.4).

5. Super-exponential gap for non-uniform μ

The last section gave an exponential gap between Q^μ and R^μ under uniform μ. Here we show that the gap can be even larger for non-uniform μ. Consider the average-case complexity of the OR-function. It is easy to see that $D^{\text{unif}}(\text{OR})$, $R^{\text{unif}}(\text{OR})$ and $Q^{\text{unif}}(\text{OR})$ are all $O(1)$, since the average input will have many ones under the uniform distribution. Now we give some examples of non-uniform distributions μ where $Q^\mu(\text{OR})$ is super-exponentially smaller than $R^\mu(\text{OR})$:

Theorem 5.1. *If $\alpha \in (0, 1/2)$ and $\mu(X) = c/\binom{N}{|X|}(|X|+1)^\alpha (N+1)^{1-\alpha}$ ($c \approx 1 - \alpha$ is a normalizing constant), then $R^\mu(\text{OR}) \in \Theta(N^\alpha)$ and $Q^\mu(\text{OR}) \in \Theta(1)$.*

Proof. Any classical algorithm for OR requires $\Theta(N/(|X|+1))$ queries on an input X. The upper bound follows from random sampling, the lower bound from a block-sensitivity argument [21]. Hence (omitting the intermediate Θs)

$$\begin{aligned} R^\mu(\text{OR}) &= \sum_X \mu(X) \frac{N}{|X|+1} \\ &= \sum_{t=0}^{N} \frac{cN^\alpha}{(t+1)^{\alpha+1}} \in \Theta(N^\alpha) \end{aligned}$$

where the last step can be shown by approximating the sum over t with an integral. Similarly, for a quantum algorithm $\Theta(\sqrt{N/(|X|+1)})$ queries are necessary and sufficient on an input X [5, 14], so

$$\begin{aligned} Q^\mu(\text{OR}) &= \sum_X \mu(X) \sqrt{\frac{N}{|X|+1}} \\ &= \sum_{t=0}^{N} \frac{cN^{\alpha-1/2}}{(t+1)^{\alpha+1/2}} \in \Theta(1). \end{aligned}$$

□

In particular, for $\alpha = 1/2 - \varepsilon$ we have the very large gap of $O(1)$ quantum versus $\Omega(N^{1/2-\varepsilon})$ classical. Note that we obtain this super-exponential gap by weighing the complexity of two algorithms (classical and quantum OR-algorithms) which are only quadratically apart on each input X. This is the phenomenon we referred to at the end of section 2.

6. General bounds for average-case complexity

In this section we prove some general bounds. First we make precise the intuitively obvious fact that if an algorithm A is faster on every input than another algorithm B, then it is also faster on average under any distribution:

Theorem 6.1. *If $\phi : \boldsymbol{R} \to \boldsymbol{R}$ is a concave function and $T_A(X) \leqslant \phi(T_B(X))$ for all X, then $T_A^\mu \leqslant \phi\left(T_B^\mu\right)$ for every μ.*

Proof. By Jensen's inequality, if ϕ is concave then $E_\mu[\phi(T)] \leqslant \phi(E_\mu[T])$, hence

$$\begin{aligned} T_A^\mu &= \sum_{X\in\{0,1\}^N} \mu(X) T_A(X) \\ &\leqslant \sum_{X\in\{0,1\}^N} \mu(X)\phi(T_B(X)) \\ &\leqslant \phi\Bigg(\sum_{X\in\{0,1\}^N} \mu(X) T_B(X)\Bigg) = \phi\left(T_B^\mu\right). \end{aligned}$$

□

In other words: taking the average cannot make the complexity-gap between two algorithms smaller. For instance, if $T_A(X) \leqslant \sqrt{T_B(X)}$ (say, A is Grover's algorithm and B is a classical algorithm for OR), then $T_A^\mu \leqslant \sqrt{T_B^\mu}$. On the other hand, taking the average *can* make the gap much larger, as we saw in theorem 5.1: the quantum algorithm for OR runs only quadratically faster than any classical algorithm on each input, but the *average-case* gap between quantum and classical can be much bigger than quadratic.

We now prove a general lower bound on R^μ and Q^μ. The classical case of the following lemma was shown in [21], the quantum case in [3]:

Lemma 6.2. *Let A be a bounded-error algorithm for some function f. If A is classical then $T_A(X) \in \Omega(bs_X(f))$, and if A is quantum then $T_A(X) \in \Omega(\sqrt{bs_X(f)})$.*

A lower bound in terms of the μ-expected block sensitivity follows:

Theorem 6.3. *For all f, μ: $R^\mu(f) \in \Omega(E_\mu[bs_X(f)])$ and $Q^\mu(f) \in \Omega(E_\mu[\sqrt{bs_X(f)}])$.*

7. Average-case complexity of MAJORITY

Here we examine the average-case complexity of the MAJORITY function. The hard inputs for MAJORITY occur when $t = |X| \approx N/2$. Any quantum algorithm needs $\Omega(N)$ queries for such inputs [3]. Since the uniform distribution puts most probability on the set of X with $|X|$ close to $N/2$, we might expect an $\Omega(N)$ average-case complexity as well. However, we will prove that the complexity is nearly $\sqrt{N}$. For this we need the following result about approximate quantum counting, which is theorem 13 of [6] (this is the forthcoming journal version of [8] and [17]; see also [18, theorem 1.10]):

Theorem 7.1 (Brassard, Høyer, Mosca, Tapp). *There exists a quantum algorithm **QCount** with the following property. For every N-bit input X (with $t = |X|$) and number of queries T, and any integer $k \geqslant 1$, **QCount** uses T queries and outputs a number $\tilde{t}$ such that*

$$|t - \tilde{t}| \leqslant 2\pi k \frac{\sqrt{t(N-t)}}{T} + \pi^2 k^2 \frac{N}{T^2}$$

with probability at least $8/\pi^2$ if $k = 1$ and probability $\geqslant 1 - 1/2(k-1)$ if $k > 1$.

Using repeated applications of this quantum counting routine we can obtain a quantum algorithm for MAJORITY that is fast on average:

Theorem 7.2. $Q^{\text{unif}}(\text{MAJ}) \in O(\sqrt{N}(\log N)^2)$.

Proof. For all $i \in \{1, \ldots, \log N\}$, define $A_i = \{X \mid N/2^{i+1} < ||X| - N/2| \leqslant N/2^i\}$. The probability under the uniform distribution of getting an input $X \in A_i$ is $\mu(A_i) \in O(\sqrt{N}/2^i)$, since the number of inputs X with k ones is $\binom{N}{k} \in O(2^N/\sqrt{N})$ for all k. The idea of our algorithm is to have $\log N$ runs of the quantum counting algorithm, with increasing numbers of queries, such that the majority value of inputs from A_i is probably detected around the ith counting stage. We will use $T_i = 100 \cdot 2^i \log N$ queries in the ith counting stage. Our MAJORITY algorithm is the following:

For $i = 1$ to $\log N$ do:

quantum count $|X|$ using T_i queries (call the estimate $\tilde{t}_i$)
if $|\tilde{t}_i - N/2| > N/2^i$, then output whether $\tilde{t}_i > N/2$ and stop.

Classically count $|X|$ using N queries and output its majority.

Let us analyse the behaviour of the algorithm on an input $X \in A_i$. For $t = |X|$, we have $|t - N/2| \in (N/2^{i+1}, N/2^i]$. By theorem 7.1, with probability $> 1 - 1/10 \log N$ we have $|\tilde{t}_i - t| \leqslant N/2^i$, so with probability $(1 - 1/10 \log N)^{\log N} \approx e^{-1/10} > 0.9$ we have $|\tilde{t}_i - t| \leqslant N/2^i$ for all $1 \leqslant i \leqslant N$. This ensures that the algorithm outputs the correct value with high probability.

We now bound the expected number of queries the algorithm needs on input X. Consider the $(i+2)$nd counting stage. With probability $1 - 1/10 \log N$ we will have $|\tilde{t}_{i+2} - t| \leqslant N/2^{i+2}$. In this case the algorithm will terminate, because

$$|\tilde{t}_{i+2} - N/2| \geqslant |t - N/2| - |\tilde{t}_{i+2} - t| > N/2^{i+1} - N/2^{i+2} = N/2^{i+2}.$$

Thus with high probability the algorithm needs no more than $i + 2$ counting stages on input X. Later counting stages take exponentially more queries ($T_{i+2+j} = 2^j T_{i+2}$), but are needed only with exponentially decreasing probability $O(1/2^j \log N)$: the probability that $|\tilde{t}_{i+2+j} - t| > N/2^{i+2}$ goes down exponentially with j precisely because the number of queries goes up exponentially. Similarly, the last step of the algorithm (classical counting) is needed only with negligible probability.

Now the expected number of queries on input X can be upper bounded by

$$\sum_{j=1}^{i+2} T_i + \sum_{k=i+3}^{\log N} T_k \cdot O\left(\frac{1}{2^{k-i-3} \log N}\right) < 100 \cdot 2^{i+3} \log N + \sum_{k=i+3}^{\log N} 100 \cdot 2^{i+3} \in O(2^i \log N).$$

Therefore under the uniform distribution the average expected number of queries can be upper bounded by $\sum_{i=1}^{\log N} \mu(A_i) O(2^i \log N) \in O(\sqrt{N}(\log N)^2)$. □

The nearly matching lower bound is:

Theorem 7.3. $Q^{\text{unif}}(\text{MAJ}) \in \Omega(\sqrt{N})$.

Proof. Let A be a bounded-error quantum algorithm for MAJORITY. It follows from the worst-case results of [3] that A uses $\Omega(N)$ queries on the hardest inputs, which are the X with $|X| = N/2 \pm 1$. Since the uniform distribution puts $\Omega(1/\sqrt{N})$ probability on the set of such X, the average-case complexity of A is at least $\Omega(1/\sqrt{N})\Omega(N) = \Omega(\sqrt{N})$. □

What about the *classical* average-case complexity of MAJORITY? Alonso *et al* [2] prove the bound $D^{\text{unif}}(\text{MAJ}) = 2N/3 - \sqrt{8N/9\pi} + O(\log N)$ for deterministic classical computers. We can also prove a linear lower bound for the *bounded-error* classical complexity, using the following lemma:

Lemma 7.4. *Let $\Delta \in \{1, \ldots, \sqrt{N}\}$. Any classical bounded-error algorithm that computes MAJORITY on inputs X with $|X| \in \{N/2, N/2 + \Delta\}$ must make $\Omega(N)$ queries on all such inputs.*

Proof. We will prove the lemma for $\Delta = \sqrt{N}$, which is the hardest case. We assume without loss of generality that the algorithm queries its input X at $T(X)$ random positions, and outputs 1 if the fraction of ones in its sample is at least $(N/2 + \Delta)/N = 1/2 + 1/\sqrt{N}$. We do not care what the algorithm outputs otherwise. Consider an input X with $|X| = N/2$. The algorithm uses $T = T(X)$ queries and should output 0 with probability at least two-thirds. Thus the probability of output 1 on X must be at most one-third, in particular

$$\Pr[\text{ at least } T(1/2 + 1/\sqrt{N}) \text{ ones in sample of size } T] \leqslant 1/3.$$

Since the T queries of the algorithm can be viewed as sampling without replacement from a set containing $N/2$ ones and $N/2$ zeros, this error probability is given by the hypergeometric distribution

$$\Pr[\text{ at least } T(1/2 + 1/\sqrt{N}) \text{ ones in sample of size } T] = \frac{\sum_{i=T(1/2+1/\sqrt{N})}^{T} \binom{N/2}{i} \cdot \binom{N/2}{T-i}}{\binom{N}{T}}.$$

We can approximate the hypergeometric distribution using the normal distribution (see e.g. [19]). Let $z_k = (2k - T)/\sqrt{T}$ and $\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-t^2/2}\, dt$, then the above probability approaches

$$\Phi(z_T) - \Phi(z_{T(1/2+1/\sqrt{N})}).$$

Note that $\Phi(z_T) = \Phi(\sqrt{T}) \to 1$ and that $\Phi(z_{T(1/2+1/\sqrt{N})}) = \Phi(2\sqrt{T/N}) \to 1/2$ if $T \in o(N)$. Thus we can only avoid having an error probability close to $1/2$ by using $T \in \Omega(N)$ queries on X with $|X| = N/2$. A similar argument shows that we must also use $\Omega(N)$ queries if $|X| = N/2 + \Delta$. □

It now follows that:

Theorem 7.5. $R^{\text{unif}}(\text{MAJ}) \in \Omega(N)$.

Proof. The previous lemma shows that any algorithm for MAJORITY needs $\Omega(N)$ queries on inputs X with $|X| \in [N/2, N/2 + \sqrt{N}]$. Since the uniform distribution puts $\Omega(1)$ probability on the set of such X, the theorem follows. □

Accordingly, *on average* a quantum computer can compute MAJORITY almost quadratically faster than a classical computer, whereas for the *worst-case* input quantum and classical computers are about equally fast (or slow).

8. Average-case complexity of PARITY

Finally we prove some results for the average-case complexity of PARITY. This is in many ways the hardest Boolean function. Firstly, $bs_X(f) = N$ for all X, hence by theorem 6.3:

Corollary 8.1. *For every μ, $R^{\mu}(\text{PARITY}) \in \Omega(N)$ and $Q^{\mu}(\text{PARITY}) \in \Omega(\sqrt{N})$.*

With high probability we can obtain an exact count of $|X|$, using $O(\sqrt{(|X| + 1)N})$ quantum queries [6]. Combining this with a μ that puts $O(1/\sqrt{N})$ probability on the set of all X with $|X| > 1$ and distributes the remaining probability arbitrarily over the X with $|X| \leqslant 1$, we obtain a distribution μ such that $Q^{\mu}(\text{PARITY}) \in O(\sqrt{N})$.

We can prove $Q^{\mu}(\text{PARITY}) \leqslant N/6$ for any μ by the following algorithm: with probability one-third output 1, with probability one-third output 0 and with probability one-third run the exact quantum algorithm for PARITY, which has worst-case complexity $N/2$ [3, 13]. This algorithm has success probability two-thirds on every input and has an expected number of queries equal to $N/6$.

More than a linear speed-up on average is not possible if μ is uniform:

Theorem 8.2. $Q^{\text{unif}}(\text{PARITY}) \in \Omega(N)$.

Proof. Let A be a bounded-error quantum algorithm for PARITY. Let B be an algorithm that flips each bit of its input X with probability one-half, records the number b of actual bitflips, runs A on the changed input Y and outputs $A(Y) + b \bmod 2$. It is easy to see that B is a bounded-error algorithm for PARITY and that it uses an *expected* number of T_A^{μ} queries on *every* input. Using standard techniques, we can turn this into an algorithm for PARITY with *worst-case* $O(T_A^{\mu})$ queries. Since the worst-case lower bound for PARITY is $N/2$ [3, 13], the theorem follows. □

Acknowledgments

We thank Harry Buhrman for suggesting this topic, and him, Lance Fortnow, Lane Hemaspaandra, Hein Röhrig, Alain Tapp and Umesh Vazirani for helpful discussions. Also thanks to Alain for sending a draft of [6]. Part of this work was done when AA visited Microsoft Research, supported by a Microsoft Research Fellowship and NSF grant CCR-9800024. RdW was partially supported by the EU Fifth Framework project QAIP, IST–1999–11234 and is also affiliated with the ILLC of the University of Amsterdam.

Appendix. Worst-case complexity of f

In this appendix we will show a lower bound of $\Omega(N)$ queries for the zero-error worst-case complexity $Q_0(f)$ of the function f on $N = n2^n$ binary variables defined in section 4. (We count binary queries this time.) Consider a quantum algorithm that makes at most T queries and that, for every X, outputs either the correct output $f(X)$ or, with probability $\leqslant 1/2$, outputs 'inconclusive'. We use the following lemma from [3]:

Lemma A.1. *The probability that a T-query quantum algorithm outputs 1 can be written as a multilinear N-variate polynomial $P(X)$ of degree at most $2T$.*

Consider the polynomial P induced by our T-query algorithm for f. It has the following properties:

(1) P has degree $d \leqslant 2T$
(2) if $f(X) = 0$ then $P(X) = 0$
(3) if $f(X) = 1$ then $P(X) \in [1/2, 1]$.

We first show that only very few inputs $X \in \{0, 1\}^N$ make $f(X) = 1$. The number of such 1 inputs for f is the number of ways to choose $k \in \{0, 1\}^n - \{0^n\}$, multiplied by the number of ways to choose $2^n/2$ independent $x_i \in \{0, 1\}^n$, which is $(2^n - 1) \cdot (2^n)^{2^n/2} < 2^{n(2^n/2+1)}$. Accordingly, the fraction of 1 inputs among all 2^N inputs X is $< 2^{n(2^n/2+1)}/2^{n2^n} = 2^{-n(2^n/2-1)}$. These X are exactly the X that make $P(X) \neq 0$. However, the following result is known [22, 23]:

Lemma A.2 (Schwartz). *If P is a non-constant N-variate multilinear polynomial of degree d, then*

$$\frac{|\{X \in \{0,1\}^N \mid P(X) \neq 0\}|}{2^N} \geqslant 2^{-d}.$$

This implies $d \geqslant n(2^n/2 - 1)$ and hence $T \geqslant d/2 \geqslant n(2^n/4 - 2) \approx N/4$. Thus we have proved that the worst-case zero-error quantum complexity of f is near maximal:

Theorem A.3. $Q_0(f) \in \Omega(N)$.

References

[1] Alon N and Spencer J H 1992 *The Probabilistic Method* (New York: Wiley)

[2] Alonso L, Reingold E M and Schott R 1997 The average-case complexity of determining the majority *SIAM J. Comput.* **26** 1–14

[3] Beals R, Buhrman H, Cleve R, Mosca M and de Wolf R 1998 Quantum lower bounds by polynomials *Proc. 39th IEEE FOCS* pp 352–61
(Beals R, Buhrman H, Cleve R, Mosca M and de Wolf R 1998 *Preprint* quant-ph/9802049)

[4] Bennett C H, Bernstein E, Brassard G and Vazirani U 1997 Strengths and weaknesses of quantum computing *SIAM J. Comput.* **26** 1510–23
(Bennett C H, Bernstein E, Brassard G and Vazirani U 1997 *Preprint* quant-ph/9701001)

[5] Boyer M, Brassard G, Høyer P and Tapp A 1998 Tight bounds on quantum searching *Fortschr. Phys.* **46** 493–505
(Boyer M, Brassard G, Høyer P and Tapp A 1996 Earlier version presented at Physcomp'96 *Preprint* quant-ph/9605034)

[6] Brassard G, Høyer P, Mosca M and Tapp A 2000 Quantum amplitude amplification and estimation *Preprint* quant-ph/0005055 (this is the forthcoming journal version of [8, 17])

[7] Brassard G, Høyer P and Tapp A 1997 Quantum algorithm for the collision problem *ACM SIGACT News (Cryptol. Column)* **28** 14–9
(Brassard G, Høyer P and Tapp A 1997 *Preprint* quant-ph/9705002)

[8] Brassard G, Høyer P and Tapp A 1998 Quantum counting *Proc. 25th ICALP (Lecture Notes in Computer Science* vol 1443) (Berlin: Springer) pp 820–31
(Brassard G, Høyer P and Tapp A 1998 *Preprint* quant-ph/9805082)

[9] Buhrman H, Dürr Ch, Heiligman M, Høyer P, Magniez F, Santha M and de Wolf R 2001 Quantum algorithms for element distinctness *Proc. 16th IEEE Conf. on Computational Complexity* pp 131–7
(Buhrman H, Dürr Ch, Heiligman M, Høyer P, Magniez F, Santha M and de Wolf R 2000 *Preprint* quant-ph/0007016)

[10] Buhrman H and de Wolf R 2001 Complexity measures and decision tree complexity: a survey *Theor. Comput. Sci.* at press

[11] Cleve R 2000 The query complexity of order-finding *Proc. 15th IEEE Conf. on Computational Complexity* pp 54–9
(Cleve R 1999 *Preprint* quant-ph/9911124)

[12] Deutsch D and Jozsa R 1992 Rapid solution of problems by quantum computation *Proc. R. Soc.* A **439** 553–8

[13] Farhi E, Goldstone J, Gutmann S and Sipser M 1998 A limit on the speed of quantum computation in determining parity *Phys. Rev. Lett.* **81** 5442–4
(Farhi E, Goldstone J, Gutmann S and Sipser M 1998 *Preprint* quant-ph/9802045)

[14] Grover L K 1996 A fast quantum mechanical algorithm for database search *Proc. 28th ACM STOC* pp 212–9
(Grover L K 1996 *Preprint* quant-ph/9605043)

[15] Hemaspaandra E, Hemaspaandra L A and Zimand M 1999 Almost-everywhere superiority for quantum polynomial time *Preprint* quant-ph/9910033

[16] Levin L A 1986 Average case complete problems *SIAM J. Comput.* **15** 285–6 (earlier version in STOC'84)

[17] Mosca M 1998 Quantum searching, counting and amplitude amplification by eigenvector analysis *MFCS'98 Workshop on Randomized Algorithms*

[18] Nayak A and Wu F 1999 The quantum query complexity of approximating the median and related statistics *Proc. 31st ACM STOC* pp 384–93
(Nayak A and Wu F 1998 *Preprint* quant-ph/9804066)

[19] Nicholson W L 1956 On the normal approximation to the hypergeometric distribution *Ann. Math. Stat.* **27** 471–83

[20] Nielsen M A and Chuang I L 2000 *Quantum Computation and Quantum Information* (Cambridge: Cambridge University Press)
[21] Nisan N 1991 CREW PRAMs and decision trees *SIAM J. Comput.* **20** 999–1007 (earlier version in STOC'89)
[22] Nisan N and Szegedy M 1994 On the degree of Boolean functions as real polynomials *Comput. Complexity* **4** 301–13 (earlier version in STOC'92)
[23] Schwartz J T 1980 Fast probabilistic algorithms for verification of polynomial identities *J. ACM* **27** 701–17
[24] Shor P W 1997 Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer *SIAM J. Comput.* **26** 1484–509 (earlier version presented at FOCS'94)
(Shor P W 1995 *Preprint* quant-ph/9508027)
[25] Simon D 1997 On the power of quantum computation *SIAM J. Comput.* **26** 1474–83 (earlier version in FOCS'94)
[26] Vitter J S and Flajolet Ph 1990 Average-case analysis of algorithms and data structures *Handbook of Theoretical Computer Science.* A: *Algorithms and Complexity* ed J van Leeuwen (Cambridge: MIT Press) pp 431–524
[27] Zalka Ch 1999 Grover's quantum searching algorithm is optimal *Phys. Rev.* A **60** 2746–51
(Zalka Ch 1997 *Preprint* quant-ph/9711070)