Quantum Search on Bounded-Error Inputs

Peter Høyer[1,⋆], Michele Mosca[2], and Ronald de Wolf[3,⋆⋆]

[1] Dept. of Computer Science, Univ. of Calgary, Alberta, Canada.
hoyer@cpsc.ucalgary.ca
[2] Dept. of Combinatorics & Optimization, Univ. of Waterloo, Ontario, Canada.
mmosca@cacr.math.uwaterloo.ca
[3] CWI. Kruislaan 413, 1098 SJ, Amsterdam, The Netherlands. rdewolf@cwi.nl

Abstract. Suppose we have n algorithms, quantum or classical, each computing some bit-value *with bounded error probability*. We describe a quantum algorithm that uses $O(\sqrt{n})$ repetitions of the base algorithms and with high probability finds the index of a 1-bit among these n bits (if there is such an index). This shows that it is not necessary to first significantly reduce the error probability in the base algorithms to $O(1/\text{poly}(n))$ (which would require $O(\sqrt{n}\log n)$ repetitions in total). Our technique is a recursive interleaving of amplitude amplification and error-reduction, and may be of more general interest. Essentially, it shows that quantum amplitude amplification can be made to work also with a *bounded-error* verifier. As a corollary we obtain optimal quantum upper bounds of $O(\sqrt{N})$ queries for all constant-depth AND-OR trees on N variables, improving upon earlier upper bounds of $O(\sqrt{N}\text{polylog}(N))$.

1 Introduction

One of the main successes of quantum computing is Grover's algorithm [10,7]. It can search an n-element space in $O(\sqrt{n})$ steps, which is quadratically faster than any classical algorithm. The algorithm assumes oracle access to the elements in the space, meaning that in unit time it can decide whether the ith element is a solution to its search problem or not. In some more realistic settings we can efficiently make such an oracle ourselves. For instance, if we want to decide satisfiability of an m-variable Boolean formula, the search space is the set of all $n = 2^m$ truth assignments, and we can efficiently decide whether a given assignment satisfies the formula. However, in these cases the decision is made without any error probability. In this paper we study the complexity of quantum search if we only have *bounded-error* access to the elements in the space.

More precisely, suppose that among n Boolean values $f_1, \ldots, f_n$ we want to find a solution (if one exists), i.e., an index j such that $f_j = 1$. For each i we have at our disposal an algorithm F_i that computes the bit f_i with two-sided error: if f_i

⋆ Supported in part by the Alberta Ingenuity Fund and the Pacific Institute for the Mathematical Sciences.

⋆⋆ This research was (partially) funded by projects QAIP (IST–1999–11234) and RESQ (IST–2001–37559) of the IST-FET programme of the EC.

J.C.M. Baeten et al. (Eds.): ICALP 2003, LNCS 2719, pp. 291–299, 2003.

is 1 then the algorithm outputs 1 with probability, say, at least 9/10, and if $f_i = 0$ then it outputs 0 with probability at least 9/10. Grover's algorithm is no longer applicable in this bounded-error setting, at least not directly, because the errors in each step will quickly add up to something uncontrollably large. Accordingly, we need to do something different to get a quantum search algorithm that works here. We will measure the complexity of our quantum search algorithms by the number of times they call the underlying algorithms F_i. Clearly, the $\Omega(\sqrt{n})$ lower bound for the standard error-less search problem, due to Bennett, Bernstein, Brassard, and Vazirani [4], also applies to our more general setting. Our aim is to give a matching upper bound.

An obvious but sub-optimal quantum search algorithm is the following. By repeating F_i $k = O(\log n)$ times and outputting the majority value of the k outcomes, we can compute f_i with error probability at most $1/100n$. If we then copy the answer to a safe place and reverse the computation to clean up (most of) the workspace, then we get something that is sufficiently "close" to perfect oracle access to the f_i bits to just treat it as such. Now we can apply Grover's algorithm on top of this, and because quantum computational errors add linearly [5], the overall difference with perfect oracle access will be negligibly small. This solves the bounded-error quantum search problem using $O(\sqrt{n}\log n)$ repetitions of the F_i's, which is an $O(\log n)$-factor worse than the lower bound. Below we will refer to this algorithm as "the simple search algorithm".

A relatively straightforward improvement over the simple search algorithm is the following. Partition the search space into $n/\log^2 n$ blocks of size $\log^2 n$ each. Pick one such block at random. We can find a potential solution (an index j in the chosen block such that $f_j = 1$, if there is such a j) in complexity $O(\log n \log\log n)$ using the simple search algorithm, and then verify that it is indeed 1 with error probability at most $1/n$ using another $O(\log n)$ invocations of F_j. Applying Grover search on the space of all $n/\log^2 n$ blocks, we obtain an algorithm with complexity $O(\sqrt{n/\log^2 n}) \cdot O(\log n \log\log n + \log n) = O(\sqrt{n}\log\log n)$.

A further improvement comes from doing the splitting recursively: we can use the improved upper bound to do the computation of the "inner" blocks, instead of the simple search algorithm. Using $T(n)$ to denote the complexity on search space of size n, this gives us the recursion

$$T(n) \leq d\left(T(\log^2 n)\sqrt{\frac{n}{\log^2 n}} + \log n\right)$$

for some constant $d > 0$. This recursion resolves to complexity $O(\sqrt{n} \cdot c^{\log^* n})$ for some constant $c > 0$. It is similar to (and inspired by) the communication complexity protocol for the disjointness problem of Høyer and de Wolf [11].

Apart from being rather messy, this improved algorithm is still not optimal. The main result of this paper is to give a relatively clean algorithm that uses the optimal number $O(\sqrt{n})$ of repetitions to solve the bounded-error search problem. Our algorithm uses a kind of "carrot-and-stick" approach that may be of more general interest. Roughly speaking, it starts with a uniform superposition of all F_i. It then amplifies all branches of the computation that give answer 1. These

branches include solutions, but they also include "false positives": branches corresponding to the 1/10 error probability of F_i's where $f_i = 0$. We then "push these back" by testing whether a 1-branch is a real positive or a false one (i.e., whether $f_i = 1$ or not) and removing most of the false ones. Interleaving these amplify and push-back steps properly, we can amplify the weight of the solutions to a constant using $O(\sqrt{n})$ repetitions. At this point we just do a measurement, see a potential solution j, and verify it classically by running F_j a few times.

As an application of our bounded-error quantum search algorithm, in Section 4 we give optimal quantum algorithms for constant-depth AND-OR trees in the query complexity setting. For any constant d, we need only $O(\sqrt{N})$ queries for the d-level AND-OR tree, improving upon the earlier $O(\sqrt{N}(\log N)^{d-1})$ algorithms of Buhrman, Cleve, and Widgerson [9]. Matching lower bounds of $\Omega(\sqrt{N})$ were already shown for such AND-OR trees, using Ambainis' quantum adversary method [1,2]. Finally, in Section 5 we indicate how the ideas presented here can be cast more generally in terms of amplitude amplification.

2 Preliminaries

Here we briefly sketch the basics and notation of quantum computation, referring to the book by Nielsen and Chuang [12] for more detail. An *m-qubit state* is a linear combination of all classical m-bit states

$$|\phi\rangle = \sum_{i \in \{0,1\}^m} \alpha_i |i\rangle,$$

where $|i\rangle$ denotes the basis state i (a classical m-bit string), the *amplitude* α_i is a complex number, and $\sum_i |\alpha_i|^2 = 1$. We view $|\phi\rangle$ as a 2^m-dimensional column vector. A measurement of state $|\phi\rangle$ will give $|i\rangle$ with probability $|\alpha_i|^2$, and the state will then collapse to the observed $|i\rangle$. A non-measuring quantum operation corresponds to applying a unitary (= linear and norm-preserving) transformation U to the vector of amplitudes. If $|\phi\rangle$ and $|\psi\rangle$ are quantum states on m and m' qubits, respectively, then the two-register state $|\phi\rangle \otimes |\psi\rangle = |\phi\rangle|\psi\rangle$ corresponds to the $2^{m+m'}$-dimensional vector that is the tensor product of $|\phi\rangle$ and $|\psi\rangle$.

The setting of query complexity is as follows. For input $x \in \{0,1\}^n$, a *query* corresponds to the unitary transformation O that maps $|i, b, z\rangle \to |i, b \oplus x_i, z\rangle$. Here $i \in [n]$ and $b \in \{0,1\}$; the z-part corresponds to the workspace, which is not affected by the query. A T-query quantum algorithm has the form $A = U_T O U_{T-1} \cdots O U_1 O U_0$, where the U_k are unitary transformations, independent of x. This A depends on x only via the T applications of O. The algorithm starts in initial all-zero state $|\mathbf{0}\rangle$ and its output (which is a random variable) is obtained from observing some dedicated part of the final superposition $A|\mathbf{0}\rangle$.

3 Optimal Quantum Algorithm for Bounded-Error Search

In this section we describe our quantum algorithm for bounded-error search. The following two facts generalize, respectively, the Grover search and the error-reduction used in the algorithms we sketched in the introduction.

Fact 1 (Amplitude amplication [8]) *Let S_0 be the unitary that puts a '-' in front of the all-zero state $|\mathbf{0}\rangle$, and S_1 be the unitary that puts a '-' in front of all basis states whose last qubit is $|1\rangle$. Let $A|\mathbf{0}\rangle = \sin(\theta)|\phi_1\rangle|1\rangle + \cos(\theta)|\phi_0\rangle|0\rangle$ where angle θ is such that $0 \leq \theta \leq \pi/2$ and $\sin^2(\theta)$ equals the probability that a measurement of the last register of state $A|\mathbf{0}\rangle$ yields a '1'. Set $G = -AS_0A^{-1}S_1$. Then $GA|\mathbf{0}\rangle = \sin(3\theta)|\phi_1\rangle|1\rangle + \cos(3\theta)|\phi_0\rangle|0\rangle$.*

Amplitude amplification is a process that is used in many quantum algorithms to increase the success probability. Amplitude amplification effectively implements a rotation by an angle 2θ in a two-dimensional space (a space different from the Hilbert space acted upon) spanned by $|\phi_1\rangle|1\rangle$ and $|\phi_0\rangle|0\rangle$. Note that we can always apply amplitude amplification regardless of whether the angle θ is known to us or not.

Fact 2 (Error-reduction) *Suppose $A|\mathbf{0}\rangle = \sqrt{p}|\phi_b\rangle|b\rangle + \sqrt{1-p}|\phi_{1-b}\rangle|1-b\rangle$, where $b \in \{0,1\}$ and $p \geq 9/10$. Then using $O(\log(1/\varepsilon))$ applications of A and majority-voting, we can build a unitary E such that $E|\mathbf{0}\rangle = \sqrt{q}|\psi_b\rangle|b\rangle + \sqrt{1-q}|\psi_{1-b}\rangle|1-b\rangle$ with $q \geq 1-\varepsilon$, and $|\psi_{b/1-b}\rangle$ possibly of larger dimension than $|\phi_{b/1-b}\rangle$ (because of extra workspace).*

We will recursively interleave these two facts to get a quantum search algorithm that searches the space $f_1, \ldots, f_n \in \{0,1\}$. We assume each f_i is computed by unitary F_i with success probability at least $9/10$. Let $\Gamma = \{j : f_j = 1\}$ be the set of solutions, and $t = |\Gamma|$ its size (which is unknown to our algorithm). The goal is to find an element in Γ if $t \geq 1$, and to output 'no solutions' if $t = 0$.

We will build an algorithm that has a superposition of all $j \in [n]$ in its first register, a growing second register that contains workspace and other junk, and a 1-qubit third register indicating whether something is deemed a solution or not. The algorithm will successively increase the weight of the basis states that simultaneously have a solution in the first register and a 1 in the third.

Consider an algorithm A that runs all F_i once in superposition, producing the state $A|\mathbf{0}\rangle$, which we rewrite as

$$\frac{1}{\sqrt{n}} \sum_{i=1}^{n} |i\rangle \left(\sqrt{p_i}|\psi_{i,1}\rangle|1\rangle + \sqrt{1-p_i}|\psi_{i,0}\rangle|0\rangle \right) = \sin(\theta)|\phi_1\rangle|1\rangle + \cos(\theta)|\phi_0\rangle|0\rangle,$$

where p_i is the probability that F_i outputs 1, the states $|\psi_{i,b}\rangle$ describe the workspace of the F_i, and $\sin(\theta)^2 = \sum_{i=1}^{n} p_i \geq 9t/10n$.

The idea is to apply a round of amplitude amplification to A to amplify the $|1\rangle$-part from $\sin(\theta)$ to $\sin(3\theta)$. This will amplify both the good states $|j\rangle|1\rangle$ for $j \in \Gamma$ and the "false positives" $|j\rangle|1\rangle$ for $j \notin \Gamma$ by a factor of $\sin(3\theta)/\sin(\theta) \approx 3$

(here we didn't write the second register). We then apply an error-reduction step to reduce the amplitude of the false positives, setting "most" of its third register to 0. These two steps together form a new algorithm that puts almost 3 times as much amplitude on the solutions as A does, and that puts less amplitude on the false positives than A. We then repeat the amplify-reduce steps on this new algorithm to get an even better algorithm, and so on.

Let us be more precise. Our algorithm will consist of a number of rounds. In round k we will have a unitary A_k that produces

$$A_k|\mathbf{0}\rangle = \alpha_k|\Gamma_k\rangle|1\rangle + \beta_k|\overline{\Gamma}_k\rangle|1\rangle + \sqrt{1-\alpha_k^2-\beta_k^2}|H_k\rangle|0\rangle,$$

where α_k, β_k are non-negative reals, $|\Gamma_k\rangle$ is a unit vector whose first register only contains $j \in \Gamma$, $|\overline{\Gamma}_k\rangle$ is a unit vector whose first register only contains $j \notin \Gamma$, and $|H_k\rangle$ is a unit vector. If we measure the first register of the above state, we will see a solution (i.e. some $j \in \Gamma$) with probability at least α_k^2. A_1 is the above algorithm A, which runs the F_i in superposition. Initially, $\alpha_1^2 \geq 9t/10n$ since each solution contributes at least $9/10n$. We want to make the good amplitude α_k grow by a factor of almost 3 in each round.

Amplitude amplification step. For each round k, define $\theta_k \in [0, \pi/2]$ by $\sin(\theta_k)^2 = \alpha_k^2 + \beta_k^2$. Applying amplitude amplification ($G_k = -A_k S_0 A_k^{-1} S_1$) gives us the state $G_k A_k|\mathbf{0}\rangle$, which we may write as

$$\frac{\sin(3\theta_k)}{\sin(\theta_k)}\alpha_k|\Gamma_k\rangle|1\rangle + \frac{\sin(3\theta_k)}{\sin(\theta_k)}\beta_k|\overline{\Gamma}_k\rangle|1\rangle + \sqrt{1-\left(\frac{\sin(3\theta_k)}{\sin(\theta_k)}\right)^2(\alpha_k^2+\beta_k^2)}|H_k\rangle|0\rangle.$$

We applied A_k twice and A_k^{-1} once, so the complexity goes up by a factor of 3.

Error-reduction step. Conditional on the qubit in the third register being 1, the error-reduction step E_k now does majority voting on $O(k)$ runs of the F_j (for all j in superposition) to decide with error at most $1/2^{k+5}$ whether $f_j = 1$. It adds one 0-qubit as the new third register and maps (ignoring its workspace, which is added to the second register)

$$E_k|j\rangle|1\rangle|0\rangle = a_{jk}|j\rangle|1\rangle|1\rangle + \sqrt{1-a_{jk}^2}|j\rangle|1\rangle|0\rangle$$
$$E_k|j\rangle|0\rangle|0\rangle = |j\rangle|0\rangle|0\rangle$$

where $a_{jk}^2 \geq 1 - 1/2^{k+5}$ if $f_j = 1$ and $a_{jk}^2 \leq 1/2^{k+5}$ if $f_j = 0$. This way, E_k removes most of the false positives.

Putting $A_{k+1} = E_k G_k A_k$ and defining α_{k+1}, β_{k+1}, $|\Gamma_{k+1}\rangle$, $|\overline{\Gamma}_{k+1}\rangle$, and $|H_{k+1}\rangle$ appropriately, we now have

$$A_{k+1}|\mathbf{0}\rangle = \alpha_{k+1}|\Gamma_{k+1}\rangle|1\rangle + \beta_{k+1}|\overline{\Gamma}_{k+1}\rangle|1\rangle + \sqrt{1-\alpha_{k+1}^2-\beta_{k+1}^2}|H_{k+1}\rangle|0\rangle.$$

Here the second register has grown by the workspace used in the error-reduction step E_k, as well as by the qubit that previously was the third register. The good amplitude has grown in the process:

$$\alpha_{k+1} \geq \alpha_k \frac{\sin(3\theta_k)}{\sin(\theta_k)} \sqrt{1 - 1/2^{k+5}}.$$

Since $x - x^3/6 \leq \sin(x) \leq x$, we have

$$\frac{\sin(3\theta_k)}{\sin(\theta_k)} \geq 3 - 9\theta_k^2/2.$$

Accordingly, as long as θ_k is small, α_k will grow by a factor of almost 3 in each round. On the other hand, the weight of the false positives goes down rapidly:

$$\beta_{k+1} \leq \beta_k \frac{\sin(3\theta_k)}{\sin(\theta_k)} \frac{1}{\sqrt{2^{k+5}}}.$$

We now analyze the number m of rounds that we need to make the good amplitude large. In general, we have $\sin(\theta_k)^2 = \alpha_k^2 + \beta_k^2$, hence $\theta_k^2 \leq 2(\alpha_k^2 + \beta_k^2)$ for the domain we are interested in. Here $\alpha_k^2 \leq 9^{k-1}\alpha_1^2$ and $\beta_k^2 \leq \frac{1}{10}(9/2^6)^{k-1}$. Note

$$\begin{aligned}
\sum_{k=1}^{m-1} \theta_k^2 &\leq 2 \sum_{k=1}^{m-1} \alpha_k^2 + \beta_k^2 \\
&\leq 2 \sum_{k=1}^{m-1} 9^{k-1}\alpha_1^2 + 2 \sum_{k=1}^{m-1} \frac{1}{10}(9/2^6)^{k-1} \\
&\leq 2 \cdot 9^{m-1}\alpha_1^2 + 1/4.
\end{aligned}$$

Therefore, m rounds of the above process amplifies the good amplitude α_k to

$$\begin{aligned}
\alpha_m &\geq \alpha_1 \prod_{k=1}^{m-1} \frac{\sin(3\theta_k)}{\sin(\theta_k)} \sqrt{1 - 1/2^{k+5}} \\
&\geq \alpha_1 \prod_{k=1}^{m-1} \left(3 - 9\theta_k^2/2\right)\left(1 - 1/2^{k+5}\right) \\
&= \alpha_1 3^{m-1} \prod_{k=1}^{m-1} \left(1 - 3\theta_k^2/2\right)\left(1 - 1/2^{k+5}\right) \\
&\geq \alpha_1 3^{m-1} \left(1 - \frac{3}{2} \sum_{k=1}^{m-1} \theta_k^2 - \sum_{k=1}^{m-1} \frac{1}{2^{k+5}}\right) \\
&\geq \alpha_1 3^{m-1} \left(1 - \frac{3}{2}(2 \cdot 9^{m-1}\alpha_1^2 + 1/4) - 1/16\right) \\
&\geq \alpha_1 3^{m-1} \left(1/2 - 3 \cdot 9^{m-1}\alpha_1^2\right).
\end{aligned}$$

In particular, whenever the (unknown) number t of solutions lies in the interval $[n/9^{m+1}, n/9^m]$, equivalently $9^m \in [n/9t, n/t]$, then we have

$$\frac{1}{3^m\sqrt{10}} \leq \sqrt{\frac{9t}{10n}} \leq \alpha_1 \leq \sqrt{\frac{t}{n}} \leq \frac{1}{3^m}.$$

This implies

$$\alpha_m \geq 0.04,$$

so the probability of seeing a solution after m rounds is at least 0.0016. By repeating this classically a constant number of times, say 1000 times, we can bring the success probability close to 1 (note to avoid confusion: these 1000 repetitions are not part of the definition of A_m itself).

The complexity C_k of the operation A_k, in terms of number of repetitions of the F_i algorithms, is given by the recursion

$$C_1 = 1 \text{ and } C_{k+1} = 3C_k + O(k),$$

where the $3C_k$ is the cost of amplitude amplification and $O(k)$ is the cost of error-reduction. This implies $C_m = O(\sum_{k=1}^{m-1} k \cdot 3^{m-k-1}) = O(3^m)$.

We now give the full algorithm when the number of solutions is unknown:

Algorithm: Quantum search on bounded-error inputs

1. for $m = 0$ to $\lceil \log_9(n) \rceil - 1$ do:
   a) run A_m 1000 times
   b) verify the 1000 measurement results, each by $O(\log n)$ runs of the corresponding F_j
   c) if a solution has been found, then output a solution and stop
2. Output 'no solutions'

This finds a solution with high probability if one exists. The complexity is

$$\sum_{m=0}^{\lceil \log_9(n) \rceil - 1} 1000 \cdot O(3^m) + 1000 \cdot O(\log n) = O(3^{\log_9(n)}) = O(\sqrt{n}).$$

If we know that there is at least one solution but we don't know how many there are, then, using a modification of our algorithm as in [7], we can find a solution using an expected number of repetitions in $O(\sqrt{N/t})$, where t is the (unknown) number of solutions. This is quadratically faster than classically, and optimal for any quantum algorithm.

4 Optimal Upper Bounds for AND-OR Trees

A d-level *AND-OR tree* on N Boolean variables is a Boolean function that is described by a depth-$d-1$ tree with interleaved ORs and ANDs on the nodes and the N input variables as leaves. More precisely, a 0-level AND-OR tree is just an

input variable, and if $f_1, \ldots, f_n$ all are d-level AND-OR trees on m variables, each with an AND (resp. OR) as root, then $\mathrm{OR}(f_1, \ldots, f_n)$ (resp. AND) is a $(d+1)$-level AND-OR tree on $N = nm$ variables. AND-OR trees can be converted easily into OR-AND trees and vice versa using De Morgan's laws, if we allow negations to be added to the tree.

Consider the two-level tree on $N = n^2$ variables with an OR as root, ANDs as its children, and fanout n in both levels. Each AND-subtree can be quantum computed by Grover's algorithm with one-sided error using $O(\sqrt{n})$ queries (we let Grover search for a '0', and output 1 if we don't find any), and the value of the OR-AND tree is just the OR of those n values. Accordingly, the construction of the previous section gives an $O(\sqrt{n} \cdot \sqrt{n}) = O(\sqrt{N})$ algorithm with *two*-sided error. This is optimal up to a constant factor [1].

More generally, for d-level AND-OR trees we can apply the above algorithm recursively to obtain an algorithm with $O(c^{d-1}\sqrt{N})$ queries. Here c is the constant hidden in the $O(\cdot)$ of the result of the previous section. For each fixed d, this complexity is $O(\sqrt{N})$, which is optimal up to a constant factor [2]. It improves upon the $O(\sqrt{N}(\log N)^{d-1})$ algorithm given in [9].

Our query complexity upper bound also implies that the minimal degree among N-variate polynomials approximating AND-OR is $O(\sqrt{N})$ [3]. Whether this upper bound on the degree is optimal remains open. The best known lower bound for the 2-level case is $\Omega(N^{1/4}\sqrt{\log N})$ [13].

5 Amplitude Amplification with Imperfect Verifier

In this section we view our construction in a more general light.

Suppose we are given some classical randomized algorithm A that succeeds in solving some problem with probability p. In addition, we are given a Boolean function χ that takes as input an output from algorithm A, and outputs whether it is a solution or not. Then, we may find a solution to our problem by repetition. We first apply algorithm A, obtaining some candidate solution, which we then give as input to the verifier χ. If χ outputs that the candidate indeed is a solution, we output it and stop, and otherwise we repeat the process by reapplying A. The probability that this process terminates by outputting a solution within the first $\Theta(\frac{1}{p})$ iterations of the loop, is lower bounded by a constant.

A quantum analogue of boosting the probability of success is to boost the *amplitude* of being in a certain subspace of a Hilbert space. Thus far, amplitude amplification [6] has assumed that we are given a perfect verifier χ: whenever a candidate solution is found, we can determine with certainty whether it is a solution or not. Formally, we model this by letting χ be computed by a deterministic classical subroutine or an exact quantum subroutine.

The main result of this paper may be viewed as an adaptation of amplitude amplification to the situation where the verifier is *not* perfect, but sometimes makes mistakes. Instead of a deterministic subroutine for computing χ, we are given a bounded-error randomized subroutine, and instead of an exact quantum subroutine, we are given a bounded-error quantum subroutine. Previously, the only known technique for handling such cases has been by straightforward

simulation of a perfect verifier: construct a subroutine for computing χ with error $\frac{1}{2^k}$ by repeating a given bounded-error subroutine of order $\Theta(k)$ times and then use majority voting. Using such direct simulations, we may construct good but sub-optimal quantum algorithms, like the $O(\sqrt{n}\log n)$ query algorithm for quantum search of the introduction. Here, we have introduced a modification of the amplitude amplification process that allows us to efficiently deal with imperfect verifiers. Essentially, our result says that imperfect verifiers are as good as perfect verifiers (up to a constant multiplicative factor in the complexity).

Acknowledgments. We thank Richard Cleve for useful discussions, as well as for hosting MM and RdW at the University of Calgary, where most of this work was done.

References

1. A. Ambainis. Quantum lower bounds by quantum arguments. In *Proceedings of 32nd ACM STOC*, pages 636–643, 2000.
2. H. Barnum and M. Saks. A lower bound on the quantum query complexity of read-once functions. quant-ph/0201007, 3 Jan 2002.
3. R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. In *Proceedings of 39th IEEE FOCS*, pages 352–361, 1998.
4. C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997.
5. E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997.
6. G. Brassard and P. Høyer. An exact quantum polynomial-time algorithm for Simon's problem. In *Proceedings of Fifth Israeli Symposium on Theory of Computing and Systems (ISTCS'97)*, pages 12–23, 1997.
7. M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4–5):493–505, 1998.
8. G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. In Lomonaco, S. J., Jr. and Brandt, H. E. (eds.): Quantum Computation and Quantum Information: A Millennium Volume. AMS Contemporary Mathematics Series, **305**:53–74, 2002.
9. H. Buhrman, R. Cleve, and A. Wigderson. Quantum vs. classical communication and computation. In *Proceedings of 30th ACM STOC*, pages 63–68, 1998.
10. L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of 28th ACM STOC*, pages 212–219, 1996.
11. P. Høyer and R. de Wolf. Improved quantum communication complexity bounds for disjointness and equality. In *Proceedings of 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS'2002)*, *Lecture Notes in Computer Science*, Vol. 2285, pages 299–310. Springer-Verlag, 2002.
12. M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
13. Y. Shi. Approximating linear restrictions of Boolean functions. Unpublished manuscript, 2002.