Electronic Colloquium on Computational Complexity, Report No. 2 (2002)

A lower bound on the quantum query complexity of read-once functions

Howard Barnum *
CCS-3, MS B256, Los Alamos National Laboratory
Los Alamos, NM 87545
barnum@lanl.gov

Michael Saks†
Dept. of Mathematics-Hill Center
110 Frelinghuysen Road
Rutgers University
New Brunswick, NJ
saks@math.rutgers.edu

November 12, 2001

Abstract

We establish a lower bound of $\Omega(\sqrt{n})$ on the bounded-error quantum query complexity of read-once Boolean functions, providing evidence for the conjecture that $\Omega(\sqrt{D(f)})$ is a lower bound for all Boolean functions. Our technique extends a result of Ambainis, based on the idea that successful computation of a function requires "decoherence" of initially coherently superposed inputs in the query register, having different values of the function. The number of queries is bounded by comparing the required total amount of decoherence of a judiciously selected set of input-output pairs to an upper bound on the amount achievable in a single query step. We use an extension of this result to general weights on input pairs, and general superpositions of inputs.

1 Introduction and summary of results

In the *quantum query* model of computation, a query register containing a string x of n bits is accessed by a quantum computer via queries. In each query, the computer may ask for a single bit i of the query register, and the value x_i of that bit is returned; queries are *quantum coherent*, which means that a computer may superpose different query requests i with complex amplitudes α_i, and is returned a superposition of the corresponding bit values x_i.

The quantum query model is the quantum analog to the classical boolean decision tree model, and is at least as powerful as the classical model. It is of great interest to compare computation in these two models, and to see the extent to which quantum computation gives an advantage over classical deterministic and randomized computation in this setting. One of the major algorithmic results in quantum computation is Grover's search algorithm [1], which can be viewed as a quantum algorithm for computing the n-bit OR function with $O(\sqrt{n})$ queries. This compares with the n queries required for deterministic decision trees and the $\Omega(n)$ queries required by classical randomized trees. This can be used to speed up brute-force search for solutions to problems (e.g. in NP)

*This work was done in part while the author was visiting DIMACS and was supported in part by NSF under grants EIA 00-80234 and 99-06105.

†Research supported by NSF grants CCR9988526, EIA 00-80234 and 99-06105.

ISSN 1433-8092

with polynomially-checkable solutions. $\Omega(\sqrt{n})$ is known to be a lower bound for OR[2] [3], perhaps our best piece of evidence that BQP $\subset$ NP.

There are two major variants of the quantum query model: the *exact* model and the *bounded error* model. In the exact model, we require that the quantum computation always output the correct answer, and in the bounded error model we allow that on any input, the computation may have a small probability ϵ of being incorrect. We write $Q_E(f)$ for the quantum complexity of f in the exact model, and $Q_\epsilon(f)$ for the quantum complexity of f in the bounded error model, where ϵ is the permissible error. (It is well known that for $\epsilon \in (0, 1/2)$ the value of ϵ only affects $Q_\epsilon(f)$ within a constant factor.) We also write $D(f)$ for the determinstic decision tree complexity of f.

In the exact model, there are examples of surprising speedups, for example the 2 bit XOR can be done exactly with one quantum query, but there are no known examples where exact quantum computation provides more than a constant factor speedup over deterministic decision trees.

In the bounded error model, the OR function provides an example where quantum computation gives a significant speedup over deterministic (and randomized) decision trees. In fact, the quadratic speedup for OR is the best speedup result known for any boolean function. Perhaps the most important problem in quantum query complexity is to resolve the following conjecture (which seems to have been suggested by several researchers):

Conjecture 1 *For any boolean function f and $\epsilon \in (0, 1/2)$, $Q_\epsilon(f) = \Omega(D(f)^{1/2})$.*

The best known result of this type says that for any f $Q_\epsilon(f) = \Omega(D(f)^{1/6})$ (This result appears in the survey article [4] and is an improvement on an earlier $\Omega(D(f)^{1/8})$ bound in [5], which is obtained by combining the arguments of [5] and an improvement, due to Nisan and Smolensky, of a result of Nisan and Szegedy[6].) It should be remarked that the conjecture is for functions whose domain is all of $\{0,1\}^n$; for functions whose domain is restricted (*promise problems*) there are much better speedups known, see e.g. [7]. In fact, the main component of Shor's factoring algorithm [8][9] is a query algorithm for the promise problem of finding the period of a function by querying its table of values(cf. [10]).

The main result of this paper is to prove the conjecture for the class of *read-once functions*, those functions expressible by a boolean formula in which each variable appears at most once. Our results provide a quantum counterpart to the lower bounds on the randomized decision tree complexity of read-once functions given in [11] and [12].

In [13], Ambainis introduced a lower bound technique for the quantum query model. He applied this technique to obtain a $\Omega(\sqrt{n})$ bound for a particular read-once function, the function which is an OR of $\sqrt{n}$ disjoint ANDs of size $\sqrt{n}$.

Our method for obtaining the $\sqrt{n}$ result generalizes Ambainis' method; in Section 3 we give a generalization of his technique. Ambainis' approach is based on a thought-experiment in which we imagine the computer to operate on a *superposition* of inputs in the query register. The idea is that successful computation of a function requires, in the thought-experiment, "decoherence" of initially coherently superposed inputs in the query register, having different values of the function. This is because successful computation must correlate input states having different values of the function with nearly orthogonal states in the part of the computer where the result is to be read. In Ambainis' main results, the inputs are, essentially, taken to be superposed with equal coefficients. The number of queries is bounded by comparing the required total amount of decoherence of a judiciously selected set of input-output pairs to an upper bound on the amount achievable in a single query step. Our result generalizes this technique to give a corresponding result using the *weighted* total decoherence of input pairs (rather than just including/excluding pairs via weights equal to zero or one), and general superpositions of inputs rather than uniform ones. We anticipate

that this result and this approach will prove useful well beyond the context of read-once functions to which we apply it in this paper.

2 Quantum query complexity

In any quantum computation model, we think of the memory of the machine as composed of *registers*, where each register has a set of allowed values. A memory configuration is an assignment of values to registers.

Each register R is associated to a complex vector space H_R whose dimension is equal to the number of allowed values of the register. H_R has a distinguished orthogonal basis whose members are in one-to-one correspondence with the possible values of the registers. We use the Dirac or "bra-ket" notation for complex vector spaces: elements of such a space are denoted by the notation $|\phi\rangle$, and viewed as complex column vectors. For such a vector $\langle\phi|$ denotes the dual row vector whose coordinates are the complex conjugates of those of $|\phi\rangle$. The notation $\langle\phi|\psi\rangle$ denotes the (complex) inner product of $|\phi\rangle$ and $|\psi\rangle$. The standard basis vectors of H_R are denoted by $|v\rangle$ where v is an allowed value for R.

A group of registers can be viewed together as a single virtual register. If $R_1, \ldots, R_k$ are registers and R is the virtual register obtained by combining them, then the value set of R is the product of the value sets of the R_i and the space H_R is naturally isomorphic to the tensor product $H_{R_1} \otimes \cdots \otimes H_{R_k}$. If $v_1, \ldots, v_k$ are possible values for $R_1, \ldots, R_k$, then $|v_1, \ldots, v_k\rangle$ is a standard basis element of H_R and is identified with $|v_1\rangle \otimes \cdots \otimes |v_k\rangle$ which is also written $|v_1\rangle|v_2\rangle \cdots |v_k\rangle$.

In particular, the entire memory can be considered as a virtual register in this way and is associated to a complex vector space H. The *quantum states* of this memory are unit vectors in H.

Now let's consider the quantum query model. Here the memory is viewed as divided into three registers: the *input register* which holds an $n+1$ bit string $x_0, x_1, \ldots, x_n$ where x_0 is fixed to 0, the *query register* which holds an integer between 0 and n, and the *auxiliary memory* which has no restrictions on its value set. The query register and auxiliary memory together comprise the *working memory*. The standard basis of the associated complex vector space H consists of vectors of the form $|x, i, z\rangle$ in which the input string is x, the query is i, and the auxiliary memory is set to z. Thus, a state of the computer is represented as:

$$\sum_{x,i,z} \alpha_{x,i,z} |x, i, z\rangle,$$

where for each memory assignment x, i, z, $\alpha_{x,i,z}$ is a complex number, and $\sum_{x,i,z} |\alpha_{x,i,z}|^2 = 1$

The space H can be viewed as the tensor product of two spaces $H_{In} \otimes H_W$ where the input space H_{In} is spanned by the 2^n basis vectors $|x\rangle$ corresponding to inputs, and the work space H_W is spanned by vectors $|q, z\rangle$ corresponding to possible contents of the working memory. The space H_W is further decomposed as $H_Q \otimes H_A$ where the query space H_Q is spanned by the $n+1$ query values $|q\rangle$ and the auxiliary space H_A is spanned by the assignments $|z\rangle$ to the auxiliary space. Thus $|x, i, z\rangle$ is identified with the tensor products $|x\rangle \otimes |i, z\rangle = |x\rangle|i, z\rangle$ and $|x\rangle \otimes |i\rangle \otimes |z\rangle = |x\rangle|i\rangle|z\rangle$.

Each computation step is a unitary operator on this vector space. In the query model, there are two types of operators allowed. A *work space* transformation is one that operates only on the work space, which means it is of the form $I_{In} \otimes A$ where I_{In} is the identity operator on H_{In} and A acts arbitrarily on H_W. The unitary operator O, called the *oracle*, operates as follows:

$$O|x, i, z\rangle = (-1)^{x_i} |x, i, z\rangle \tag{1}$$

An algorithm is specified by (1) an arbitrary sequence $U_1, \ldots, U_t$ of work space operators and (2) a pair of orthogonal projectors P_0 on the space H_W, i.e., a pair of linear maps satisfying $(P_0)^2 = P_0$, $P_1^2 = P_1$ and $P_0 + P_1 = I_W$.

An algorithm is executed as follows. The memory is initialized in the basis state with the input register set to the input x and all other registers set to 0. Then the sequence $U_1, O, U_2, O, \ldots, U_t, O$ is applied to the computer. For $l \in \{1, \ldots, t\}$ the pair U_l, O is called the l^{th} step of the computation. Observe that the operations U_l and O leave the input register unchanged. Formally this means that the state of the computer is always of the form $|x\rangle \otimes |\Psi\rangle$ where x is the input and $|\Psi\rangle$ is a vector of H_W (generally not a standard basis state).

The output of the computation is either 0 or 1, determined according to the following probability distribution. If the final state of the computation is $|x\rangle \otimes |\Psi\rangle$ then the computation outputs j with probability equal to $\|P_j|\Psi\rangle\|^2$. (Note that the definition of P_0 and P_1 guarantees that the vectors $P_0|\Psi\rangle$ and $P_1|\Psi\rangle$ are orthogonal and sum to a unit vector, which implies that the two probabilities sum to 1.) The process which generates this distribution is called a *measurement*.

Variants of this model have been considered; in particular the oracle O can be replaced by a more general transformation that transforms the work space depending on the value of the input bit indexed by the query register. It is well known that this generality can not speed up the computation by more than a factor of 2.

The complexity of the algorithm is measured by the number of calls t to the oracle. In the bounded-error quantum query model, we fix some $\epsilon < 1/2$ and a computation is considered to successfully compute f if it *ϵ-computes* f, which means that for every input, the probability that the algorithm gives the wrong answer for that input, is no greater than ϵ. The ϵ-error quantum query complexity of f, denoted $Q_\epsilon(f)$ is the minimum number of steps in an algorithm that ϵ-computes f. It is well known that the choice of $\epsilon \in (0, 1/2)$ only affects the complexity up to a constant factor.

To avoid confusion, we note that a different way of phrasing quantum query complexity problems is sometimes used: the oracle is said to be a "black box function," g. This g is what we are calling the "input," (and using the letters x or y for); it is *not* the function f being computed. The black box terminology sometimes calls our f something like P, and P is said to be a *property* of the black box function g. This model does not represent the input g as a state of a register in the computer. Rather, the computer is our computer minus the input register, and a query step is an application of the unitary O_g to the computer state. Generally, O_g is viewed as acting on two registers, an "input" register which is homologous to what we have called the query register, and an output register. Its action in the standard basis is to compute g of the state in the input register, and write it (in modular arithmetic to ensure unitarity) in the output register, while keeping the input around. (An alternative phase version of the query unitary, similar to (1), is sometimes used in this picture, too.) We mention this approach primarily to forestall any confusion that could arise because the terms "function" and "input" may used for different things on this approach than on the one we have adopted.

3 A general lower bound on quantum query complexity of Boolean functions

In this section, we present a general extension of Ambainis' lower bound approach.

Let f be an n-variate boolean function whose query complexity we want to lower bound. The lower bound is expressed in terms of a complex vector $|\alpha\rangle$ of length 2^n indexed by inputs (so it is

a member of H_{In}) and a $2^n \times 2^n$ nonnegative real matrix Γ indexed by pairs of inputs, satisfying $\Gamma_{xy} = 0$ if $f(x) = f(y)$. For such a matrix Γ, for each $i \in \{1, \ldots, n\}$ we define for $x \in \{0,1\}^n$:

$$\nu_{x,i} = \sum_{y: x_i \neq y_i} \Gamma_{xy}, \tag{2}$$

the total *weight* of inputs differing from x on variable i. Further, for $i \in \{1, \ldots, n\}$ and $b \in \{0,1\}$ we define:

$$\begin{aligned} \nu_i^b &= \max_{x: f(x)=b} \nu_{x,i} \\ \nu_i &= \nu_i^0 \nu_i^1 \\ \nu &= \max_{j \in \{1,\ldots,n\}} \nu_j. \end{aligned}$$

The main result of this section is:

Theorem 1 *Let f be an n-variate boolean function. Let $|\alpha\rangle$ be a nonnegative real valued vector indexed by $\{0,1\}^n$ and Γ be a nonnegative real matrix indexed by $\{0,1\}^n \times \{0,1\}^n$ satisfying $\Gamma_{x,y} = 0$ whenever $f(x) = 0$ or $f(y) = 1$. If there is a quantum algorithm that ϵ-computes f using t queries, then*

$$t \geq \frac{\langle\alpha|\Gamma|\alpha\rangle(1 - 2\sqrt{\epsilon(1-\epsilon)})}{\sqrt{\nu}} = \Omega\left(\frac{\langle\alpha|\Gamma|\alpha\rangle}{\sqrt{\nu}}\right). \tag{3}$$

Buhrman and Szegedy (personal communication) have independently obtained a similar result.

This should be compared to Theorem 6 of [13]. In this theorem, Ambainis gives a lower bound which can be obtained from the above theorem by letting Γ be a 0-1 matrix and letting $|\alpha\rangle$ be a 0-1 vector. Thus $|\alpha\rangle$ is the characteristic function of some subset Z of inputs and Γ is the characteristic function of some relation R on $f^{-1}(1) \times f^{-1}(0)$. (Actually, if we define $X = Z \cap f^{-1}(1)$ and $Y = Z \cap f^{-1}(1)$, Ambainis' defines the vector $|\alpha\rangle$ to have $\alpha_z = 1/\sqrt{|X|}$ for $z \in X$ and $1/\sqrt{|Y|}$ for $z \in Y$, but this normalization does not affect the bounds.) This choice of Γ and α leads to simplifications of both the numerator and denominator.

When specialized as above, the denominator in our expression reduces to the denominator in Ambainis' theorem 6. Ambainis defines $l_{x,i}$ to be the number of $y \in Y$ such that $R(x,y)$ and $x_i \neq y_i$, and $l_{y,i}$ similarly. Our parameter ν specializes to Ambainis' parameter l_{max} which is defined as the maximum of the product $l_{x,i} l_{y,i}$, over ones x, zeroes y, and query indices i.

Ambainis also defines $m(m')$ as the minimum over $x \in X$ ($y \in Y$) of the number of $y' \in Y$ ($x' \in X$) such that $R(x,y')$ ($R(y,x')$), which bounds the numerator from below. Then Ambainis' Theorem 6 says that the complexity is $\Omega(\sqrt{mm'/l_{max}})$.

In the application to read-once functions in the next section, we will not need the full generality of Theorem 1. In particular, as Ambainis does, we restrict Γ to be the characteristic function of a relation. On the other hand, the nonuniformity of the coefficients α will be crucial to our results.

The remainder of this section is devoted to a proof of Theorem 1. The proof is a more or less straightforward generalization of Ambainis' bound.

Let $U_1, \ldots, U_t$ be a sequence of work space operators and P_0, P_1 be a pair of orthogonal projectors on H_W that specify an algorithm. Once we fix an algorithm, then on input x, the state of the computation after j steps is of the form $|x\rangle|\Psi_x(j)\rangle$, where $|\Psi_x(j)\rangle \in H_W$. (We will normally suppress the index j). Let us consider the set of vectors $\{|\Psi_x = \Psi_x(t)\rangle : x \in \{0,1\}^n\}$ after t computational steps, but before the final measurement.

Proposition 1 (Ambainis) *For a computation to ϵ-compute f, it is necessary that for any x, y such that $f(x) = 1$, $f(y) = 0$,*

$$|\langle \Psi_x | \Psi_y \rangle| \leq 2\sqrt{\epsilon(1-\epsilon)} \; . \tag{4}$$

Proof: If the computation ϵ-computes f, then

$$\begin{aligned} ||P_0|\Psi_x\rangle||^2 =: \eta_x \leq \epsilon, \\ ||P_1|\Psi_y\rangle||^2 =: \eta_y \leq \epsilon \; . \end{aligned} \tag{5}$$

Now,

$$\begin{aligned} |\langle \Psi_x | \Psi_y \rangle| &= |\langle \Psi_x | P_1 | \Psi_y \rangle + \langle \Psi_x | P_0 | \Psi_y \rangle| \\ &\leq |\sqrt{\langle \Psi_x | P_1 | \Psi_x \rangle \langle \Psi_y | P_1 | \Psi_y \rangle} + \sqrt{\langle \Psi_x | P_0 | \Psi_x \rangle \langle \Psi_y | P_0 | \Psi_y \rangle}| \\ &= \sqrt{(1-\eta_x)\eta_y} + \sqrt{\eta_x(1-\eta_y)} \leq 2\sqrt{\epsilon(1-\epsilon)} \; , \end{aligned} \tag{6}$$

by the Schwarz inequality and (5). ■

We remark that these necessary conditions are not sufficient [**?**].

Define M to be the matrix with elements $M_{xy} = |\langle \Psi_x | \Psi_y \rangle|$. (When we want to explicitly consider the situation at step l, we write $M(l)$ for the matrix with elements $|\langle \Psi_x(l) | \Psi_y(l) \rangle|$.) It is useful to group the inputs according to whether $f(x) = 0$ or 1, and view the matrix as a two-by-two matrix of block structure $M_{0,0}$, $M_{0,1}$, $M_{1,0}$, $M_{1,1}$ given by this grouping. Proposition 1 involves only the off-diagonal block, say, $M_{1,0}$. The general approach of Ambainis involves looking at how much a single query can decrease the matrix elements of this off-diagonal block. Of course, many matrix elements must be considered at once, because any individual matrix element M_{xy} can be brought down to zero by a single query to any bit i for which $x_i \neq y_i$. In fact, such a query will reduce to zero *all* M_{xy} such that $x_i \neq y_i$. However, such a query will fail to have any impact on matrix elements for which $x_i = y_i$. There is thus a tradeoff between various sets of matrix elements. A successful deterministic classical algorithm must cannily choose i's, depending on the results of previous queries, such that each query distinguishes many inputs that were not distinguished by previous ones. For a probabilistic classical algorithm, at each query probability may be distributed between the indices i; and in a quantum algorithm, complex *amplitude* rather than probability is distributed over the query indices. But in each case there is a tradeoff: more probability, or more amplitude, on a query that distinguishes one set of input pairs, can reduce the probability, or amplitude, on queries distinguishing another set.

In order to incorporate such tradeoffs while providing a necessary condition for ϵ-computation less complicated than the full set of conditions implied by Proposition 1, we might consider averaging all the off-diagonal-block matrix elements' moduli $|M_{xy}|$. Since they must all drop below $\kappa := 2\sqrt{\epsilon(1-\epsilon)}$, so must their average. In fact, we may consider any desired positive weighted sum S of the off-diagonal-block matrix elements, $\sum_{x,y} \mu_{xy} |M_{xy}|$. For reasons that are still a bit mysterious to us, it turns out that it is useful to express the weight $\mu_{xy} = \Gamma_{xy} \alpha_x \alpha_y$, where Γ is a nonnegative real matrix and $|\alpha\rangle$ is a unit vector with nonnegative entries. On the face of it, this more complex expression provides no additional generality, but it provides additional flexibility in the analysis. The vector $|\alpha\rangle$ can be interpreted, as Ambainis does, as an initial superposition of inputs in the query register.

As an immediate consequence of Proposition 1 we have:

Proposition 2 *Let f be an n-variate boolean function, and let A be a t-step quantum query algorithm that attempts to compute f. Let $|\alpha\rangle$ be a unit vector indexed by $\{0,1\}^n$ with nonnegative real entries and Γ be a matrix indexed by $\{0,1\}^n \times \{0,1\}^n$ with nonnegative real entries that satisfies $\Gamma_{x,y} = 0$ if $f(x) = f(y)$. If A ϵ-computes f then:*

$$\sum_{xy} \Gamma_{xy}\alpha_x\alpha_y|M(t)_{xy}| \leq \langle\alpha|\Gamma|\alpha\rangle\sqrt{2\epsilon(1-\epsilon)}\,. \tag{7}$$

For $l \in \{0,\ldots,t\}$, let us define

$$S_l = \sum_{xy} \Gamma_{xy}\alpha_x\alpha_y|M(l)_{xy}|. \tag{8}$$

Since $M(0)$ is the all 1 matrix, Proposition 2 implies:

Proposition 3 *For a t-step computation to ϵ-compute f, it must be the case that*

$$S_0 - S_t \geq \sum_{xy} \Gamma_{xy}|\alpha_x||M_{xy}||\alpha_y|(1-2\sqrt{\epsilon(1-\epsilon)})\,. \tag{9}$$

We will now get a lower bound on t by upper bounding $S_l - S_{l+1}$, the amount that the sum can decrease as the result of a single query.

Proposition 4

$$S_l - S_{l+1} \leq 2\sqrt{\nu}\,. \tag{10}$$

The proof of this proposition appears in an Appendix.

If we multiply this upper bound on decrease per query by t, this must exceed the difference $S_0 - S_t$. This together with Proposition 3 completes the proof of the theorem.

4 Read-once Boolean functions

A read-once Boolean function is one which can be written as a formula in propositional logic, involving each variable x_i (each bit of the input string) only once. Each such function can be represented by an AND/OR tree. This is a rooted labeled tree having n leaves, each corresponding to a different variable (with some possibly negated), and where each internal node is labeled either AND or OR. Each AND (resp., OR) node in the tree is associated to a function which is defined recursively as the AND (resp. OR) of the functions computed by its children.

Without loss of generality, we may assume that all of the children of an AND node are OR nodes, and vice versa. Also, we restrict attention to monotone functions, which are those whose leaves are all nonnegated variables, since the query complexity of the function is preserved under negation of variables.

In this section, we use the convention that the variable x indicates zeroes of the function, and the variables y and z indicate ones of the function.

Theorem 2 *$\Omega(\sqrt{n})$ is a lower bound on the bounded-error quantum query complexity of all read-once Boolean functions.*

We outline the proof technique before providing details.

We will apply Theorem 1. For this we need to define the matrix Γ and the vector $|\alpha\rangle$. We identify a subset of $\{0,1\}^n$ called *critical* inputs. These are, intuitively, the inputs on which f is hardest to compute. (The same notion of critical input plays a similar role in the lower bound proofs for the randomized query complexity of read-once functions [11] [12]).

We also define what it means for two critical inputs $x \in f^{-1}(0)$ and $y \in f^{-1}(1)$ to be *neighbors*; intuitively these are pairs of inputs that are hard to distinguish. We define the matrix Γ to be the characteristic function of the neighbor relation on the set of critical inputs. Given these choices, it will turn out from the definition of critical inputs that $\sqrt{\nu}$ is always one.

The main work of the proof comes in choosing the vector $|\alpha\rangle$. We look for a choice of the vector $|\alpha\rangle$ that maximizes the expression in the lower bound of Theorem 1 (given our particular choice of Γ). This (continuous) maximization problem is formulated using Lagrange multipliers, and gives rise to a set of first-order conditions. We then construct $|\alpha\rangle$ that satisfies the first order conditions.

This solution is constructed inductively. Assume that the root is an AND and has r children and for $i \in \{1,\ldots,r\}$ let g^i denote the function computed at child i. (The case that the root is an OR involves only obvious minor alterations.) We write n_i for the number of (boolean) variables in g_i. Thus $n := \sum_{i=1}^{r} n_i$ is the number of (boolean) variables in f. Assume that we have determined $|\alpha^i\rangle$ for each of the g_i. We construct $|\alpha\rangle$ in terms of these. Further we show that if $|\alpha^i\rangle$ gives a bound of $\kappa\sqrt{n_i}$ for each of the g_i, then $|\alpha\rangle$ gives a bound of $\kappa\sqrt{n}$ for f.

We proceed with the detailed proof. We have the read-once function f represented by tree T, and express f as $g^1 \wedge \ldots \wedge g^r$ where g^i are the functions computed at the children of the root labeled by AND.

In choosing a Γ and $|\alpha\rangle$ for applying Theorem 1, we will focus our attention on *critical inputs*. An input is critical if for each AND node, at most one child evaluates to 0 and for each OR node, at most one child evaluates to 1. A critical input in $f^{-1}(1)$ is a *critical one* and a critical input in $f^{-1}(0)$ is a *critical zero*.

We write X (resp. Y) for the set of critical zeros (resp., critical ones) of f. For $i \in \{1,\ldots,r\}$ we write X^i (resp., Y^i) for the set of critical zeros (resp., critical ones) of g^i. We use the letter x to denote an element of X, the letters y and z to denote elements of Y. Also x^i denotes an element of X^i and y^i and z^i denote elements of Y^i.

Observe that since the root is an AND, a critical one y may be written in the form $y = y^1 \ldots y^r$, where for each j, y^j is a critical one of g^j. For a critical zero x, exactly one of the children of the root evaluates to 0. We say that x is of *type* i, for $i \in \{1,\ldots,r\}$, if g^i evaluates to 0. A critical zero x of type i may be written in the form $x = z^1 \ldots z^{i-1} x^i z^{i+1} \ldots z^r$, where x^i is a critical zero of g_i and for $j \neq i$, z^j is a critical one of g^j.

Let $x \in X$ and $y \in Y$ and let i be the type of x. We say that $y = y^1 \ldots y^r$ and $x = z^1 \ldots z^{i-1} x^i z^{i+1} \ldots z^r$, are *neighbors* provided that $z^j = y^j$ for $j \neq i$ and x^i and y^i are *neighbors* (defined recursively). We denote by R the neighbor relation on $X \times Y$ and R_i the neighbor relation on $X_i \times Y_i$. It is easy to see that two neighbors differ on exactly one input variable and consequently, for any critical input w and any $j \in \{1,\ldots,n\}$, w has at most one neighbor that differs from it on variable j.

When we apply Theorem 1 we take Γ to be the characteristic function of the relation R. It is easily seen that for this Γ, the parameter ν appearing in the denominator in Theorem 1 is just 1: by the last sentence of the previous paragraph, the quantity $\nu_{w,j}$ is at most 1 for any critical input w and $j \in \{1,\ldots,n\}$.

Having fixed Γ, we now want to choose $|\alpha\rangle$. Without loss of generality we will take the coordinates of $|\alpha\rangle$ to be nonnegative real numbers and assume that they are zero outside of $X \cup Y$. We

look for an $|\alpha\rangle$ such that the lower bound expression in Theorem 1 is maximum. This means we want to solve:

$$\begin{aligned} &\max \sum_{(x,y)\in R} \alpha_x \alpha_y \\ \text{S.T.} \quad &\sum_x \alpha_x^2 + \sum_y \alpha_y^2 = 1 \end{aligned} \tag{11}$$

From Lagrange multiplier optimization for the above problem, we get the first-order conditions (FOCs):

$$\begin{aligned} \alpha_x &= \mathcal{C} \sum_{y:(x,y)\in R} \alpha_y \ , \\ \alpha_y &= \mathcal{C} \sum_{x:(x,y)\in R} \alpha_x \ . \end{aligned} \tag{12}$$

Here $\mathcal{C}$ is a constant to be determined.

Suppose we find $\mathcal{C}$ and a unit vector $|\alpha\rangle$ satisfying (12). If we multiply the first FOC by α_x and sum on $x \in X$ we get that the objective function is equal to $\frac{1}{\mathcal{C}} \sum_{x\in X} \alpha_x^2$. Similarly if we multiply the second FOC by α_y and sum on $y \in Y$ we have that the objective function equals $\frac{1}{\mathcal{C}} \sum_{y\in Y} \alpha_y^2$. This implies that $\sum_{x\in X} \alpha_x^2 = \sum_{y\in Y} \alpha_y^2 = \frac{1}{2}$ and the value of the objective function is $\frac{1}{2\mathcal{C}}$. We will prove:

Lemma 1 *There is a nonnegative real unit vector $|\alpha\rangle$ satisfying (12) with $\mathcal{C} = 1/\sqrt{n}$.*

Theorem 2 now follows immediately from the lemma and Theorem 1

The proof of the lemma is by induction. For the base case, we take f to be the univariate functions $f(x_1) = x_1$. For this function $X = \{0\}$, $Y = \{1\}$ and $\alpha_0 = \alpha_1 = \frac{1}{\sqrt{2}}$ solves the FOC with $\mathcal{C} = 1$. For the induction step, we assume that the lemma holds for each of the functions g_i and prove that it holds for f.

Let $x = y^1 \dots y^{i-1} x^i y^{i+1} \dots y^r$ be an element of X. All neighbors y of x must have a critical one y^i in the i-th place that is a neighbor of x^i, while agreeing with x in the other places, and thus be of the form:

$$y^1 \dots y^i \dots y^r \ . \tag{13}$$

So,

$$\alpha_x \equiv \alpha_{y^1 \dots x^i \dots y^r} = \mathcal{C} \sum_{y^i:(x^i,y^i)\in R_i} \alpha_{y^1 .. y^i \dots y^r}. \tag{14}$$

Similarly, for $y = y^1 \dots y^r \in Y$,

$$\alpha_y \equiv \alpha_{y^1 \dots y^r} = \mathcal{C} \sum_i \sum_{x^i:(x^i,y^i)\in R_i} \alpha_{y^1 \dots y^{i-1} x^i y^{i+1} \dots y^r} \ . \tag{15}$$

By the induction hypothesis, for each $i \in \{1, \dots, r\}$, we have a unit vector α^i that satisfies the first-order conditions for g^i:

$$\alpha^i_{y^i} = \mathcal{C}_i \sum_{x^i:R_i(x^i,y^i)} \alpha^i_{x^i} \ , \tag{16}$$

$$\alpha^i_{x^i} = \mathcal{C}_i \sum_{y^i:R_i(x^i,y^i)} \alpha^i_{y^i} \ . \tag{17}$$

with $\mathcal{C}_i = \frac{1}{\sqrt{n}}$.

We proceed to establish the induction step. We guess that the weights at the top level are the product of the weights at the next level down, up to a constant which can depend on the *type* of the input whose weight we are computing. (Here, only the x's have distinct types, depending on which of the g_i has the value zero.)

$$\alpha_y \equiv \mathcal{A}\alpha_{y^1 \ldots y^r} = \alpha^1_{y^1}\alpha^2_{y^2}\ldots\alpha^r_{y^r} \tag{18}$$

$$\alpha_x \equiv \alpha_{y^1 \ldots x^i \ldots y^r} = \mathcal{B}_i\alpha^1_{y^1}\alpha^2_{y^2}\ldots\alpha^i_{x^i}\ldots\alpha^r_{y^r}\ . \tag{19}$$

We check these guesses by plugging them into both sides of (14) and (15), respectively, obtaining:

$$(\alpha_x =)\mathcal{B}_i\alpha_{y^1 \ldots x^i \ldots y^r} = \mathcal{C}\mathcal{A}\alpha^1_{y^1}\ldots\alpha^{i-1}_{y^{i-1}}\ldots\left(\sum_{y^i:(x^i,y^i)\in R_i}\alpha^i_{y^i}\right)\alpha^{i+1}_{y^{i+1}}\ldots\alpha^r_{y^r}\ , \tag{20}$$

$$(\alpha_y =)\mathcal{A}\alpha^1_{y^1}\ldots\alpha^r_{y^r} = \mathcal{C}\sum_i \mathcal{B}_i\alpha^1_{y^1}\ldots\alpha^{i-1}_{y^{i-1}}\left(\sum_{x^i:R_i(x^i,y^i)}\alpha^i_{x^i}\right)\alpha^{i+1}_{y^{i+1}}\ldots\alpha^r_{y^r}\ . \tag{21}$$

The parenthesized sums in these expressions evaluate to $\alpha^i_{x^i}/\mathcal{C}_i$ and $\alpha^i_{y^i}/\mathcal{C}_i$, respectively, from the lower level Lagrange FOCs (16-17). So our guess solves the higher level FOCs. In the equation (20) for w_x, this requires

$$\mathcal{B}_i = \frac{\mathcal{C}\mathcal{A}}{\mathcal{C}_i}\ . \tag{22}$$

In the equation (21) for w_y, we obtain:

$$w_y = \mathcal{C}\sum_i \mathcal{B}_i\frac{1}{\mathcal{C}_i}w_{y^1}\ldots w_{y^i}\ldots w_{y^r}\ , \tag{23}$$

and (since the weight product on the RHS is i-independent), this requires (substituting for $\mathcal{B}_i$ using (22))

$$\mathcal{A} = \mathcal{C}\sum_i \frac{\mathcal{C}\mathcal{A}}{\mathcal{C}_i^2}\ , \quad \text{i.e.,} \quad \frac{1}{\mathcal{C}^2} = \sum_i \frac{1}{\mathcal{C}_i^2}\ . \tag{24}$$

Since $\mathcal{C}_i = 1/\sqrt{n_i}$ we deduce $\mathcal{C} = 1/\sqrt{\sum_i n_i} = 1/\sqrt{n}$ as required.

References

[1] Lov Grover, "A fast quantum mechanical algorithm for database search," *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC)*, pp. 212–219, May 1998.

[2] C. H. Bennett, G. Brassard, E. Bernstein, and U. Vazirani, "Strengths and weaknesses of quantum computing," *SIAM Journal on Computing*, vol. 26, pp. 1510–1523, 1997.

[3] L. Grover, "How fast can a quantum computer search?," 1998, arXiv.org e-print quant-ph/9809029.

[4] H. Buhrman and R. de Wolf, "Complexity measures and decision tree complexity," *Theoretical Computer Science*, to appear. (Available at `http://www.cwi.nl/ rdewolf/`).

[5] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf, "Quantum lower bounds by polynomials," *FOCS '98*, pp. 352–361, 1998.

[6] N. Nisan and M. Szegedy, "On the degree of boolean functions as real polynomials," *Computational Complexity*, pp. 301–313, 1994.

[7] D. Simon, "On the power of quantum computation," *SIAM J. Comp.*, vol. 26, pp. 1474–1483, 1997.

[8] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," *Proc. 37th ann. symp. on the foundations of computer science*, pp. 56–65, 1994.

[9] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comp.*, pp. 1484–1509, 1997.

[10] R. Cleve, "The query-complexity of order-finding," 1999, arXiv.org e-print quant-ph/9911124.

[11] M. Saks and A. Wigderson, "Probabilistic Boolean decision trees and the complexity of evaluating game trees," *Proceedings of the 27th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pp. 29–38, 1986.

[12] M. Santha, "On the Monte-Carlo Boolean decision tree complexity of read-once formulae," *Proceedings of the 6th IEEE Structure in Complexity Theory*, pp. 180–187, 1991.

[13] A. Ambainis, "Quantum lower bounds by quantum arguments," *Proceedings of the 32nd Annual ACM Symposium on the Theory of Computing (STOC)*, pp. 636–643, 2000.

[14] M. Szegedy, H. Barnum and M. Saks "Quantum decision trees and semidefinite programming," submitted, 2001.

A Proof of Proposition 4

Without loss of generality for our purposes, we may take Γ to be symmetric, upper triangular, or lower triangular. We proceed to establish an upper bound on the magnitude of the decrease of the weighted sum (8) in a single query. It will be convenient to assume Γ is symmetric. Define $|\Psi_x^i\rangle$ as the component $|\Psi_x\rangle$ having i in the query register, so that

$$|\Psi_x\rangle = \sum_i |\Psi_x^i\rangle \ . \tag{25}$$

Then

$$M = \sum_i M^i \tag{26}$$

where M^i is defined via $(M^i)_{xy} = \langle\Psi_x^i|\Psi_y^i\rangle$. (Just write M_{xy} as an inner product, use (25), and note that the $i \neq j$ terms are zero.) To reduce clutter, define $\rho_{xy}^i := \alpha_x^* M_{xy}^i \alpha_y$. In this notation we want to upper bound $\sum_i \sum_{xy} \Gamma_{xy} |\rho_{xy}^i|$. We consider the inner sum first. We first upper bound $|\rho_{x,y}^i|$ as a linear combination of ρ_{xx}^i and ρ_{yy}^i. For this purpose we introduce a nonnegative matrix $\beta = \beta_{xy}$, whose entries we will specify later. We have:

$$|\rho_{xy}^i| \leq \sqrt{\rho_{xx}^i \rho_{yy}^i} = \sqrt{\rho_{xx}^i \frac{1}{\beta_{xy}} \beta_{xy} \rho_{yy}^i} \leq \frac{1}{2}(\beta_{xy}\rho_{xx}^i + \frac{1}{\beta_{xy}}\rho_{yy}^i) \ . \tag{27}$$

(The first inequality is due to the positivity of ρ^i, which requires that the determinant of any principal minor be positive; the second is the arithmetic-geometric mean inequality). Then

$$\sum_{xy:x_i\neq y_i} \Gamma_{xy}|\rho^i_{xy}|$$
$$\leq \sum_x \sum_{y:y_i\neq x_i} \Gamma_{xy}\frac{1}{2}(\beta^i_{xy}\rho^i_{xx} + \frac{1}{\beta^i_{xy}}\rho^i_{yy}) = \sum_x \sum_{y:y_i\neq x_i} \Gamma_{xy}\beta^i_{xy}\rho^i_{xx} \ . \tag{28}$$

The last equality is just due to the symmetry under $x \leftrightarrow y$.

We now define

$$\beta^i_{xy} = \sqrt{\frac{\nu_{y,i}}{\nu_{x,i}}} \ , \tag{29}$$

where $\nu_{x,i}$ was defined at the beginning of this section.

The last expression in (28) becomes:

$$\sum_x \rho^i_{xx} \sum_{y:y_i\neq x_i} \Gamma_{xy}\sqrt{\frac{\nu_{yi}}{\nu_{xi}}} \ . \tag{30}$$

Recalling the definition of ν^b_i at the beginning of the section, we can bound this expression by:

$$\begin{aligned}
\sum_x \rho^i_{xx} \sum_{y:y_i\neq x_i} \Gamma_{xy}\sqrt{\frac{\nu_i^{1-x_i}}{\nu_{xi}}} &\leq \sum_x \rho^i_{xx}\sqrt{\nu_{xi}\nu_i^{1-x_i}} \\
&\leq \sum_x \rho^i_{xx}\sqrt{\nu_i^0\nu_i^1} \\
&= \sum_x \mathrm{tr}\ \rho^i\sqrt{\nu_i} \\
&\leq \sum_x \mathrm{tr}\ \rho^i\sqrt{\nu}
\end{aligned}$$

Summing on i then yields:

$$\sum_i \sum_{xy:x_i\neq y_i} \Gamma_{xy}|\rho^i_{xy}| \leq \sqrt{\nu} \tag{31}$$

This equation is an important lemma, which we use to establish our bound on the decrease of the weighted sum in a single query. For quantities which change in a query, we distinguish the post-query quantity by priming it.

$$\begin{aligned}
S' - S &:= \sum_{xy} \Gamma_{xy}(|\rho_{xy}| - |\rho'_{xy}|) = \sum_{xy} \Gamma_{xy}(|\rho_{xy} - \rho'_{xy}|) \\
&\leq \sum_i \sum_{xy} \Gamma_{xy}(|\rho^i_{xy} - \rho'^i_{xy}|)
\end{aligned} \tag{32}$$

We can bound the term in parentheses by noting that since ρ^i is the input register density matrix relative to i in the query index the query multiplies density matrix elements ρ^i_{xy} by a factor $(-1)^{x_iy_i}$, leaving them unchanged if $x_i = y_i$. Thus we obtain

$$\sum_i \sum_{xy:x_i\neq y_i} \Gamma_{xy}(|\rho^i_{xy} - \rho'^i_{xy}|) \leq \sum_i \sum_{xy:x_i\neq y_i} \Gamma_{xy}|\rho^i_{xy}| + |\rho'^i_{xy}|. \tag{33}$$

We can then apply Eq. (31) to each of these terms, obtaining:

$$S - S' \leq \sqrt{\nu} \sum_i (\mathrm{tr}\ \rho + \mathrm{tr}\ \rho') = 2\sqrt{\nu}\ . \tag{34}$$

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'