Quantum query complexity and semi-definite programming

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

Michael Saks†
Dept. of Mathematics–Hill Center
Rutgers University
110 Frelinghuysen Road
Piscataway, NJ, 08854
saks@math.rutgers.edu

Mario Szegedy ‡
Dept. of Computer Science
Rutgers University
110 Frelinghuysen Road
Piscataway, NJ, 08854
szegedys@cs.rutgers.edu

Abstract

We reformulate quantum query complexity in terms of inequalities and equations for a set of positive semidefinite matrices. Using the new formulation we: 1. show that the workspace of a quantum computer can be limited to at most $n + k$ qubits (where n and k are the number of input and output bits respectively) without reducing the computational power of the model. 2. give an algorithm that on input the truth table of a partial boolean function and an integer t runs in time polynomial in the size of the truth table and estimates, to any desired accuracy, the minimum probability of error that can be attained by a quantum query algorithm attempts to evaluate f in t queries. 3. use semidefinite programming duality to formulate a dual SDP $\hat{P}(f, t, \epsilon)$ that is feasible if and only if f can not be evaluated within error ϵ by a t-step quantum query algorithm Using this SDP we derive a general lower bound for quantum query complexity that encompasses a lower bound method of Ambainis and its generalizations. 4. Give an interpretation of a generalized form of branching in quantum computation.

1. Introduction and summary of results

The bounded-error quantum query model is both relevant to understanding powerful explicit non-query quantum algorithms such as Shor's factoring algorithm [19],[20] and theoretically important as the quantum analogue of the classical decision tree model. Quadratic speedups over the best probabilistic decision trees have been shown for some functions (notably OR of n variables with $O(\sqrt{n})$ queries [9], [10], for which there is a matching lower bound [7]) and examples also exist where only a linear speedup is possible. For partial functions, examples of exponential speedups exist, but for total functions, the gap is known to be at most a degree-6 polynomial [6], while the best gap that has been established for an explicit function is quadratic. It is suspected that the 6th-degree bound on the gap can be considerably strengthened. Many other interesting questions about quantum query complexity, involving both lower and upper bounds, remain open as well.

One barrier to better understanding of the quantum query model is the lack of simple mathematical representations of quantum computations. While classical query complexity (both deterministic and randomized) has a natural intuitive description in terms of decision trees, there is no such easy description of quantum query complexity. The main difference between the classical and quantum case is that classical computations branch into noninteracting subcomputations (as represented by the tree) while in quantum computations, because of the possibility of destructive interference between subcomputations, there is no obvious analog of branching.

In this paper, we provide a mathematical representation of quantum query complexity of a (partial) boolean function as a *semidefinite programming (SDP) feasibility problem*. More specifically, given a partial boolean function f, integer t and $\varepsilon \in [0, 1]$ we construct a semidefinite program $P(f, t, \varepsilon)$ that is feasible if and only if f can be computed within error ε by a t-step quantum query algorithm.

Using this characterization we obtain the following:

- show that the workspace of a quantum computer can be limited to at most $n + k$ qubits (where n and k are the number of input and output bits respectively) without reducing the computational power of the model (see Theorem 1). This improves upon the best previous (unpublished) bound of $O(n \log n)$, due to Nayak [15].
- give an algorithm that on input the truth table of a partial boolean function and an integer t runs in time polynomial in the size of the truth table and estimates, to

* This work was done in part while the author was visiting DIMACS and AT&T Research, and was supported by NSF grant EIA-00-80234 and the US DOE.
†Research supported by NSF grant CCR-9988526 and EIA-00-80234.
‡Research supported by NSF grant CCR-0105692.

Proceedings of The 18th IEEE Annual Conference on Computational Complexity
ISBN 0-7695-1879-6/03 $17.00 © 2003 IEEE

any desired accuracy, the minimum probability of error that can be attained by a quantum query algorithm attempts to evaluate f in t queries. (see Section 6). Previously, Aaronson [1] had obtained bounds that were subexponential for space bounded query complexity.

- use semidefinite programming duality to formulate a dual SDP $\hat{P}(f,t,\epsilon)$ that is feasible if and only if f can not be evaluated within error ϵ by a t-step quantum query algorithm (see Section 8). Using this SDP we derive a general lower bound for quantum query complexity (Theorem 4) that encompasses a lower bound method of Ambainis [3, 4] and its generalizations (e.g., [5]).

- Give an interpretation of a generalized form of branching in quantum computation (section 7).

The SDP $P(f,t,\epsilon)$ is constructed roughly as follows. Given a quantum query algorithm for a function whose input is a set $S \subseteq \{0,1\}^n$, one can define for each time step j and input $x \in S$, the vector $\Psi_x^{(j)}$ which is the quantum state of the machine at step j on input x. This vector can be written in the form $\sum_{i=0}^{n} |i\rangle \Psi_{x,i}^{(j)}$ where the index i represents the value of the next query to be asked. For each i,j, the indexed system of vectors $(\Psi_{x,i}^{(j)} : x \in S)$ is associated with its $S \times S$ *Gram matrix* $M_i^{(j)}$, whose x,y entry is the inner product $\langle \Psi_{x,i}^{(j)} \mid \Psi_{y,i}^{(j)} \rangle$. Each of these matrices is positive semidefinite and the rules for the execution of a quantum query algorithm are translated into linear equations in the entries of these matrices. The requirement that a t-step quantum algorithm evaluate the function f within error ϵ is translated into a requirement that the sum $M^{(t)} = \sum_{i=0}^{n} M_i^{(t)}$ of the final Gram matrices be decomposable as a sum $\sum_{z \in T} \Gamma_z$ where T is the output domain of the function and Γ_z is a positive semidefinite matrix whose diagonal entries $\Gamma[x,x]$ corresponding to $x \in f^{-1}(z)$ are at least $1-\epsilon$. The constraint system we obtain on the matrices $(M_i^{(j)})$ and (Γ_z) defines the SDP $P(f,t,\epsilon)$.

The idea of representing a quantum query algorithm by looking at the evolution of the Gram matrices associated with the system of possible state vectors originates with Ambainis [3, 4]. He used this together with a necessary condition on the matrix $M^{(t)}$ to develop a lower bound method for quantum query algorithms. Independently of our work, Kitaev [13], developed a similar semidefinite programming formulation of quantum communication complexity which he used to prove limitations on the robustness of quantum coin flipping algorithms.

2. Preliminaries

2.1. Finite functions

In a query complexity model (classical or quantum), we wish to compute a function $f : S \longrightarrow T$ where:

- $S \subseteq \Sigma^n$ with Σ finite and n a positive integer,
- T is a finite set.

An input to f is a point $x = (x_1, \ldots, x_n) \in \Sigma^n$. The set S is the *input domain*, Σ is the *variable domain*, and T is the *output domain*. We say that f is *boolean* if $\Sigma = \{0,1\}$, f is a *decision function* if $T = \{0,1\}$, and f is *total*, which means $S = \Sigma^n$. The most commonly studied class of functions are total boolean decision function.

In this paper all functions are assumed to be boolean. We do not restrict to total functions or to decision functions.

2.2. Hilbert Spaces

The essential feature of quantum computation models is that at all times the memory state is described as a point in a Hilbert space, that is, an inner product space over the complex numbers. Here we review some basic notation for such spaces.

For a finite set W, H_W denotes the complex vector space with W as a distinguished orthonormal basis. An element Ψ of H_W is typically viewed as a column vector with entries Ψ_w for $w \in W$. For $w \in W$, we use Dirac notation, $|w\rangle$, to denote the unit basis vector associated with w. More generally, in this notation, an arbitrary vector in H_W is denoted by $|\Psi\rangle$, but for vectors other than those in the standard basis we will generally omit the $|\rangle$, and write simply Ψ.

The inner product of vectors Ψ and Φ, denoted $\langle \Psi | \Phi \rangle$, is $\langle \Psi | \Phi \rangle := \sum_k \overline{\Psi_k} \Phi_k$, where $\overline{\Psi_k}$ is the complex conjugate of Ψ_k.

2.3. Operators and matrices

Since, for finite sets W_1 and W_2, the spaces H_{W_1} to H_{W_2} have distinguished bases, we identify a linear transformation $A : H_{W_1} \longrightarrow H_{W_2}$ with its matrix representation with respect to these bases. As a matrix A has rows indexed by W_1 and columns indexed by W_2, so we refer to A as a $W_1 \times W_2$ matrix, and its entries are denoted $A[w_1, w_2]$.

In particular if A maps $H = H_W$ to itself (is an *operator* on H) then the associated matrix is a square $W \times W$ matrix. We write $A^\top$ for the transpose of A, and $A^\dagger$ for the *adjoint* of A, which is the matrix obtained by replacing each entry of $A^\top$ by its conjugate. An operator A on space H is:

1. *Hermitian* if $A = A^\dagger$,

Proceedings of The 18th IEEE Annual Conference on Computational Complexity
ISBN 0-7695-1879-6/03 $17.00 © 2003 IEEE

2. *positive semidefinite* (resp. *positive definite*) if $\langle \Psi \mid A\Psi \rangle \geq 0$ (resp. > 0) for all $\Psi \in H$.

3. *negative semidefinite* (resp. *negative definite*) if $-A$ is positive semidefinite (resp. *positive definite*).

4. *unitary* if $\langle \Psi \mid \Phi \rangle = \langle A\Psi \mid A\Phi \rangle$ for all $\Psi, \Phi \in H$,

5. a *projector* if $A^2 = A$.

Note that a unitary operator is invertible.

Two operators A and B are *orthogonal* if $\langle A\Psi \mid B\Phi \rangle = 0$ if for all $\Psi, \Phi \in H_W$. A *complete system of orthogonal projectors (CSOP)* is an indexed set $(P_z : z \in T)$ of projectors that are pairwise orthogonal and satisfy $\sum_{z \in T} P_i = I_H$ where I is the identity operator on H. If $H = \oplus_{z \in T} H_z$ is a direct sum decomposition of H into pairwise orthogonal subspaces then there is a unique CSOP $(P_z : z \in T)$ such that P_z is the identity on H_z and maps $H_{z'}$ to 0 for all $z' \neq z$.

2.4. Gram matrices

Let $(\Psi_x : x \in S) \subseteq H$ be an indexed family of vectors lying in the Hilbert space H. The *Gram matrix* Gram$(\Psi_x : x \in S)$ associated to the family is the $S \times S$ matrix M with entries given by:

$$M[x, y] = \langle \Psi_x \mid \Psi_y \rangle.$$

It is easy to see that M is positive semidefinite and Hermitian and rank$(M) \leq \dim(H)$. Conversely,

Proposition 1 *Let S be a finite set and let M be a $S \times S$ positive semidefinite Hermitian matrix of rank r. Then for any Hilbert space H of dimension at least r, there is an indexed family $(\Psi_x : x \in S) \subseteq H$ whose Gram matrix is M.*

Two such representations of the same matrix are connected by a unitary transformation:

Proposition 2 *Let H be a finite dimensional Hilbert space and $(\Psi_x : x \in S) \subseteq H$ and $(\Phi_x : x \in S) \subseteq H$. Then Gram$(\Psi_x : x \in S)$ = Gram$(\Phi_x : x \in S)$ if and only if there is a unitary operator U on H satisfying $U\Psi_x = \Phi_x$, for each $x \in S$.*

Suppose $(\Psi_x : x \in S) \subseteq H$ has Gram matrix M and $(P_z : z \in T)$ is a CSOP for H corresponding to the orthogonal decomposition $\oplus_{z \in T} H_z$. Then the collection $(M_z : z \in T)$ where M_z is the Gram matrix of $(P_z \Psi_x : x \in S) \subseteq H_z$ is easily seen to satisfy $M = \sum_z M_z$. The following proposition says that if we are given an arbitrary decomposition $M = \sum_{z \in T} N_z$ of M as a sum of Hermitian positive semidefinite matrices with rank$(N_z) \leq \dim(H_z)$ we can apply a unitary transformation to $(\Psi_x : x \in S)$ so that each N_z is the Gram matrix of the P_z-projection of $(U\Psi_x : x \in S)$:

Lemma 1 *Let H be a Hilbert space and $(H_z : z \in T)$ be a set of pairwise orthogonal subspaces such that $H = \oplus_{z \in T} H_z$ and let $(P_z : z \in T)$ be the associated CSOP. Let $(\Psi_x : x \in S) \subseteq H$ have Gram matrix M and let $(N_z : z \in T)$ be a family of Hermitian positive semidefinite matrices such that $M = \sum_z N_z$ and let rank$(N_z) \leq \dim(H_z)$. Then there is a unitary transformation U on H such such that for each $z \in T$, $N_z =$ Gram$(P_z U \Psi_x : s \in S)$.*

Proof: By proposition 1, for each z there is a family of vectors $(\Phi_{z,x} : x \in T) \subseteq H_z$ whose Gram matrix is M_z. Define $\Phi_x = \sum_{z \in T} \Phi_{z,x}$. Then Gram$(\Phi_x : x \in S)$ is $\sum_z M_z = M$ and so by proposition 2 there is a unitary transformation U that maps Ψ_x to Φ_x for each $x \in S$, and this transformation fulfills the requirements of the theorem. ■

2.5. Quantum query complexity

The quantum query complexity model is one of many models of quantum computation, all of which have certain common features, which we review first. In such a model, there is a memory consisting of registers, where each register has a set of allowed values. A *memory configuration* is an assignment of values to registers. Each register R has a finite set $V(R)$ of possible values and is associated with the complex vector space $H_{V(R)}$ (as defined in Section 2.2), which we denote H_R for simplicity.

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 single virtual register in this way and is associated to a complex vector space H. A unit vector in H is called a *quantum state* of the memory.

In the *quantum query model*, the memory is viewed as divided into three registers:

- the *input register*, which holds the input $x \in \{0, 1\}^n$,
- the *query register*, which holds an integer between 0 and n, and
- the *working memory*, which has no a priori restrictions on the value set.

Proceedings of The 18th IEEE Annual Conference on Computational Complexity
ISBN 0-7695-1879-6/03 $17.00 © 2003 IEEE

The query register and working memory together make up the *accessible memory*.

The standard basis of the associated complex vector space H consists of vectors of the form $|x, i, w\rangle$ in which x is the input, i is the value of the query register and w is the value of the working memory. Thus, a state of the memory is a vector of the form:

$$|\beta\rangle = \sum_{x,i,w} \beta_{x,i,w}|x, i, w\rangle,$$

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

The space H is thus $H_{input} \otimes H_A$, where H_{input} is the *input space* spanned by $\{|x\rangle : x \in \{0,1\}^n\}$, and the *accessible space* H_A is $H_Q \otimes H_W$, where the *query space* H_Q is spanned by vectors $\{|i\rangle : i \in \{0, \ldots, n\}\}$, and the *work space* H_W is spanned by vectors $|w\rangle$ where w ranges over the value set of the working memory. Thus $|x, i, w\rangle$is identified with the tensor products $|x\rangle|i, w\rangle$ and $|x\rangle|i\rangle|w\rangle$.

The space H has a natural decomposition as a direct sum of the mutually orthogonal subspaces $(|x\rangle \otimes H_A : x \in S)$. Thus an arbitrary element Ψ of H_A can be written uniquely in the form:

$$\Psi = \sum_{x \in S} |x\rangle \Psi_x,$$

where $\Psi_x \in H_A$. Similarly, the space H_A has a natural decomposition as a direct sum of the mutually orthogonal subspaces $(|i\rangle \otimes H_W : 0 \leq i \leq n)$ and thus $\Psi_x \in H_A$ can be uniquely written in the form:

$$\Psi_x = \sum_{i=0}^{n} |i\rangle \Psi_{x,i}$$

where $\Psi_{x,i} \in H_W$.

It is, of course, customary to think of the working memory as composed of bits. In the quantum model, these bits are referred to as *qubits*, and the number b of qubits in the working memory (which is the base 2 logarithm of the size of the value set) is the natural measure of space for this model. This parameter b completely specifies the space H.

In any quantum computation model, a computation step is a unitary operator on H that is applied to the current memory state. One of the features that defines a model is the set of unitary operators that are allowed as computation steps. In the quantum query model, this set consists of a single fixed operator O called the *oracle operator* and a family of operators called *accessible space operators*. The oracle operator O is defined by its action on the standard basis:

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

Application of O to the memory state is referred to as a *query* to the input. For $i = 0$, x_i is not part of the input and x_0 is defined to be the constant 0. Thus, $i = 0$ acts as a switch to "turn off" the action of O, and is called a *null query*. (Null queries are needed in the model for an important technical reason; without them the model is not even capable of computing the identity function from $\{0, 1\}$ to $\{0, 1\}$.)

An *accessible space operator* is an operator Q on the accessible space H_A. Q acts on the entire state $H = H_{input} \otimes H_A$ by interpreting it as $I_{input} \otimes Q$ where I_{input} is the identity on H_{input}.

A quantum query algorithm (QQA) with n input variables and output domain T is specified by:

- The number b of qubits in the working memory.
- The *query stage*, which is a sequence $(U^{(j)} : 0 \leq j \leq t)$ of unitary accessible space operators.
- The *output stage*, which is a set of accessible space operators $(P_z : z \in T)$ comprising a complete set of of orthogonal projectors

The QQA is executed as follows. On input x, the memory is initialized in the state $U^{(0)}|x, 0, 0\rangle$. Then the sequence $O, U^{(1)}, O, \ldots, O, U^{(t-1)}, O, U^{(t)}$ is applied to the state. For $j \in \{1, \ldots, t\}$, step j of the query stage consists of the jth application of O followed by the application of $U^{(j)}$. Note that the oracle operator and any accessible space operator leave the input register unchanged. Formally, this means that the action of the oracle operator and of an accessible space operator is invariant on each subspace $|x\rangle \otimes H_A$ and thus on input x, the state Υ of the computer remains of the form $|x\rangle\Upsilon_x$ where x is the input and $\Upsilon_x \in H_A$ changes over time. Because of this invariance, for each $x \in S$, we can define the operator O_x on H_A so that $O|x\rangle\Upsilon_x = |x\rangle O_x \Upsilon_x$. [1] The action of O_x on $\Upsilon_x \in H_A$ is described explicitly by decomposing Υ_x as $\sum_{i=0}^{n} |i\rangle|\Upsilon_{x,i}\rangle$ and defining:

$$O_x|i\rangle|\Upsilon_{x,i}\rangle(-1)^{x_i}|i\rangle|\Upsilon_{x,i}\rangle \quad (2)$$

The state of the memory during an execution of the query stage of a QQA on input x is described by the sequence $(\Psi_x^{(j)} : 0 \leq j \leq t)$ of vectors in H_A defined by:

[1] Another variant of the quantum query model defines the "oracle" unitaries O_x as we have just done, and omits all mention of our "input register." When this variant is used, sometimes the string x is viewed as the table of values of a function, and the contents of what we have called the query register may be termed the "input" to this function, the oracle O_x is a "black box" for computing the function; this "black-box" function should *not* be confused with the function f which our algorithm computes (which in this variant of the model is often called a "property" of the black-box function, in the common case that it is Boolean), nor should the inputs to this function be confused with what we called inputs above). We mention this to forestall confusion for readers who may encounter, or may have already encountered, this variant in the literature.

Proceedings of The 18th IEEE Annual Conference on Computational Complexity
ISBN 0-7695-1879-6/03 $17.00 © 2003 IEEE

$$\Psi_x^{(j)} = \begin{cases} U^{(0)}|0\rangle|0\rangle & \text{if } j = 0 \\ U^{(j)} O \Psi_x^{(j-1)} & \text{if } j \geq 1. \end{cases} \tag{3}$$

The output of the QQA on input x is a random element Z of T chosen according to a probability distribution that depends on the final state $\Psi_x^{(t)}$, where for $z \in T$, the probability $\pi_x(z)$ that $Z = z$ on input x is given by:

$$\pi_x(z) = ||P_z \Psi_x^{(t)}||^2.$$

The completeness and orthonormality of the P_z and normalization of $\Psi_x^{(t)}$ guarantee that $\sum_{z \in T} \pi_x(z) = 1$.

This process whereby the output is determined from the final state depending on a CSOP is called *projective measurement*. There are alternative definitions of the model which restrict or generalize the type of measurement allowed, but these do not change the model in a substantive way. One alternative is to allow more general *POVM measurements* [16] for determining the output, but this does not increase the power of the model. Another alternative is to hypothesize that the accessible memory includes a distinguished output register whose value set is T, and to restrict the final measurement to "reading" that register, which means that P_z is the projection (with respect to the standard basis) onto the subspace where the output register is set to z. This is not a real restriction since given an arbitrary CSOP the final unitary $U^{(t)}$ can be used to rotate the space so that the CSOP is transformed into this special one.

Given a function $f : S \longrightarrow T$ and $\varepsilon \in [0, 1/2)$, we say that an algorithm *computes f within error ε* provided that on any input x, the probability that the algorithm outputs $f(x)$ is at least $1 - \varepsilon$, i.e.,

$$\pi_x(f(x)) \geq 1 - \varepsilon. \tag{4}$$

The complexity of a QQA is the number t of queries (applications of the oracle O). The minimum t for query algorithms satisfying (4) for f is the *ϵ-error quantum query complexity of f*, denoted $\mathrm{QQC}_\varepsilon(f)$. Since every deterministic query algorithm (decision tree) can be implemented as a QQA, $\mathrm{QQC}_\varepsilon(f) \leq n$ where n is the number of variables of f. For $0 < \varepsilon < \delta < 1/2$,

$$\mathrm{QQC}_\delta(f) \leq \mathrm{QQC}_\varepsilon(f) \leq C(\varepsilon, \delta)\mathrm{QQC}_\delta(f),$$

where $C(\varepsilon, \delta)$ is independent of the function f. Thus, up to a constant factor, the value of $\varepsilon \in (0, 1/2)$ is irrelevant. In contrast, the special case of $\varepsilon = 0$, known as *exact* or *zero-error* query complexity, may differ from the *bounded-error* case of $\varepsilon \in (0, 1/2)$ by a factor that depends on n.

3. Representing quantum query complexity by semidefinite programs

Given a partial boolean function f, integer t and $\varepsilon \in [0, 1/2)$ we are interested in the question:

> Is there a t-step QQA that computes f within error ε?

In this section we show that for each f, t, ε one can construct a system $P(f, t, \varepsilon)$ of equalities and inequalities in a set of real variables in such a way that the above question has an affirmative answer if and only if $P(f, t, \varepsilon)$ has a nonempty solution set, i.e., is feasible. The system of constraints $P(f, t, \varepsilon)$ belongs to the class of *semidefinite programs*.

We start by considering a t-step QQA, $(U^{(j)} : 0 \leq j \leq t)$ and $(P_z : z \in T)$ that computes f within error ε. We associate to the QQA two indexed families of matrices $(M_i^{(j)} : 0 \leq j \leq t-1, 1 \leq i \leq n)$ and $(\Gamma_z : z \in T)$ and a matrix $M^{(t)}$, each with rows and columns indexed by S.

We will show (Theorem 1) that this system of matrices satisfies the system of constraints $P(f, t, \varepsilon)$, and conversely, if the system of constraints $P(f, t, \varepsilon)$ is feasible then there is a t-step QQA that computes f within error ε.

First assume a t-step QQA is given as above. For each $x \in S$, (3) associates to the QQA the sequence $(\Psi_x^{(j)} : 0 \leq j \leq t)$ of vectors in $H_A = H_Q \otimes H_W$. Thus for each $x \in S$ and $0 \leq j \leq t$ in the we can write:

$$\Psi_x^{(j)} = \sum_{i=0}^{n} |i\rangle \Psi_{x,i}^{(j)} \tag{5}$$

where $\Psi_{x,i}^{(j)} \in H_W$.

Now for $0 \leq j \leq t$, $0 \leq i \leq n$, and $z \in T$, we define the following $S \times S$ matrices:

$$M^{(j)} = \mathrm{Gram}(\Psi_x^{(j)} : x \in S). \tag{6}$$

$$M_i^{(j)} = \mathrm{Gram}(\Psi_{x,i}^{(j)} : x \in S). \tag{7}$$

$$\Gamma_z = \mathrm{Gram}(P_z \Psi_x^{(j)} : x \in S). \tag{8}$$

We refer to the matrices $M^{(j)}$ as *state Gram matrices* and Γ_z as *output Gram matrices*.

We now define the system of constraints $P(f, t, \varepsilon)$ on the matrices $(M_i^{(j)} : 0 \leq j \leq t-1, 0 \leq i \leq n)$, $M^{(t)}$, and $(\Gamma_z : z \in T)$. The following fixed matrices are useful in expressing the system.

- E denotes the constant 1 matrix.
- $\mathbf{0}$ denotes the constant 0 matrix.

Proceedings of The 18th IEEE Annual Conference on Computational Complexity
ISBN 0-7695-1879-6/03 $17.00 © 2003 IEEE

- $\mathbf{E_i}$ for $0 \leq i \leq n$ is the matrix with entries $E_i[x,y] = (-1)^{x_i+y_i}$.
- Δ_z for $z \in T$ denotes the diagonal matrix with diagonal entries $\Delta_z[x,x] = 1$ if $f(x) = z$ and $\Delta_z[x,x] = 0$ otherwise.
- F denotes the matrix with $F[x,y] = 1$ if $f(x) \neq f(y)$ and $F[x,y] = 0$ if $f(x) = f(y)$.

We also define two partial orders and one binary operation on the set of $S \times S$:

- $A \geq B$ means $A[x,y] \geq B[x,y]$ for all $x, y \in S$,
- $A \succeq B$ means $A - B$ is positive semidefinite,
- $A * B$ is the matrix with entries $A * B[x,y] = A[x,y]B[x,y]$.

Semidefinite program $P(f,t,\varepsilon)$. Find $S \times S$ real symmetric positive semidefinite matrices $M^{(t)}$, $(M_i^{(j)} : 0 \leq j \leq t-1, 1 \leq i \leq n)$ and $(\Gamma_z : z \in T)$ satisfying:

$$\sum_{i=0}^{n} M_i^{(0)} = E \tag{9}$$

$$\sum_{i=0}^{n} M_i^{(j)} = \sum_{i=0}^{n} E_i * M_i^{(j-1)} \quad \text{for } 1 \leq j \leq t-1 \tag{10}$$

$$M^{(t)} = \sum_{i=0}^{n} E_i * M_i^{(t-1)} \tag{11}$$

$$M^{(t)} = \sum_{z \in T} \Gamma_z \tag{12}$$

$$\Delta_z * \Gamma_z \geq (1-\varepsilon)\Delta_z. \tag{13}$$

Theorem 1 *Let $f : S \longrightarrow T$ be a partial boolean function with domain $S \subseteq \{0,1\}^n$. Let t be a natural number and $\varepsilon \geq 0$. There is a t-step QQA that computes f within error ε if and only if $P(f,t,\varepsilon)$ is feasible. Furthermore, given a feasible solution to $P(f,t,\varepsilon)$ there is a QQA where the number r of distinct values needed for the working memory is at most the maximum of $\{\max(\mathit{rank}(M_i^{(j)})) : 0 \leq i \leq n, 0 \leq j \leq t-1)\}$ and $\sum_{z \in T} \mathit{rank}(\Gamma_z)$. In particular, $r \leq |S||T|$ and therefore there is an algorithm whose work space has at most $\lceil \log |S| + \log |T| \rceil$ qubits.*

Proof: We first assume as above that we have a t-step QQA that computes f within error ε and consider the matrices defined by (7) and (8) and the matrix $M^{(t)}$ defined by (6). These matrices are positive semidefinite because they are Gram matrices. We show that they satisfy (9),(10),(11),(12) and (13). They need not be real-valued or symmetric, but we achieve this by replacing each matrix A in the system by $A + A^\top$ which preserves all the other restrictions and thus gives a feasible solution to $P(f,t,\epsilon)$.

We now verify conditions (9), (10) and (11). Even though the matrices $(M^{(j)} : 0 \leq j \leq t-1)$ are not explicitly part of $P(f,t,\varepsilon)$ we continue to write $M^{(j)} = \sum_{i=0}^{n} M_i^{(j)}$ for $0 \leq j \leq t-1$. We have that $M^{(0)} = E$ since it is the Gram matrix of an indexed set of vectors all of whose members are the same unit vector; thus (9) holds. It follows from (5), (6) and (7) that for $1 \leq j \leq t$, $\Psi_x^{(j)} = U^{(j)} O \Psi_x^{(j-1)}$. Therefore, Proposition 1 implies that $(\Psi_x^{(j)} : x \in S)$ and $(O\Psi_x^{(j-1)} : x \in S)$ have the same Gram matrix. The Gram matrix of the former is $M^{(j)}$ while that of the latter has entries:

$$\begin{aligned}
\langle O\Psi_x^{(j-1)} \mid O\Psi_y^{(j-1)} \rangle &= \\
\langle \sum_{i=0}^{n} (-1)^{x_i} |i\rangle \Psi_{x,i}^{(j-1)} \mid \sum_{i=0}^{n} (-1)^{y_i} |i\rangle \Psi_{y,i}^{(j-1)} \rangle &= \\
\sum_{i=0}^{n} (-1)^{x_i+y_i} \langle \Psi_{x,i}^{(j)} \mid \Psi_{y,i}^{(j)} \rangle &= \\
\sum_{i=0}^{n} M_i^{(j-1)} * E_i[x,y], & \qquad (14)
\end{aligned}$$

which gives (10) and (11).

Constraint (13) follows immediately from condition (4) that the algorithm computes f within ε. To show (12), we use the properties of systems of orthogonal projectors to write:

$$\begin{aligned}
M^{(t)}[x,y] &= \langle \Psi_x^{(t)} \mid \Psi_y^{(t)} \rangle \\
&= \langle \sum_{z \in T} P_z \Psi_x^{(t)} \mid \sum_{z \in T} P_z \Psi_y^{(t)} \rangle \\
&= \sum_{z \in T} \langle P_z \Psi_x^{(t)} \mid P_z \Psi_y^{(t)} \rangle \\
&= \sum_{z \in T} \Gamma_z[x,y].
\end{aligned}$$

This completes the proof that the existence of a t-step QQA that computes f within error ε gives a feasible solution for $P(f,t,\varepsilon)$. We now turn to the converse. We assume that we have a feasible solution to $P(f,t,\varepsilon)$ and construct from it a t-step QQA for f within error ε, whose working memory has value set $[r] = \{0,\ldots,r-1\}$ (as required by the final part of the theorem). We define H_W to be the the free Hilbert space of dimension r spanned by $[r]$. The query register has value set $\{0,\ldots,n\}$ which spans the $n+1$ dimensionsal space H_Q and so the accessible space $H_A = H_Q \otimes H_W$ has dimension nr.

Proceedings of The 18th IEEE Annual Conference on Computational Complexity
ISBN 0-7695-1879-6/03 $17.00 © 2003 IEEE

We need to construct a sequence $U^{(0)}, \ldots, U^{(t)}$ unitary operators acting on H_A and a complete system of orthonormal projectors $(P_z : z \in T)$ also acting on H_A such that the associated algorithm computes f within error ε.

In the proof of the first part of the theorem, we used a given QQA to define for each $i \in \{0, \ldots, n\}$ and $j \in \{0, \ldots, t\}$ an indexed family of vectors $(\Psi_{x,i}^{(j)} : x \in S)$ and used these families with the operators $(P_z : z \in T)$ to construct a solution to $P(f, t, \varepsilon)$. Now, we work backwards. We will use the solution of $P(f, t, \varepsilon)$ to construct the families of vectors, and then use the families to construct the QQA $(U^{(j)} : 0 \leq j \leq t)$ and $(P_z : z \in T)$.

By proposition 1, we can find an indexed set of vectors $(\Psi_x^{(t)} : x \in S) \subset H_A$ whose Gram matrix is $M^{(t)}$ and for each $0 \leq i \leq n, 1 \leq j \leq t-1$, we can find an indexed set of vectors $(\Psi_{x,i}^{(j)} : x \in S) \subseteq H_W$ whose Gram matrix is $M_i^{(j)}$. For $1 \leq j \leq t-1$, let $\Psi_x^{(j)} = \sum_{i=0}^{n} |i\rangle \Psi_{x,i}^{(j)}$; for each such j, the Gram matrix of $(\Psi_x^{(j)} : x \in S)$ is $M^{(j)} = \sum_i M_i^{(j)}$. Since $M^{(0)} = E$, all of the vectors $\Psi_x^{(0)}$ are the same unit vector in H_A and we define $U^{(0)}$ to be a unitary operator on H_A mapping $|0\rangle|0\rangle$ to $\Psi_x^{(0)}$.

For $1 \leq j \leq t$, from (14), the Gram matrix of $(O\Psi_x^{(j-1)} : x \in S)$ is $\sum_{i=0}^{n} M^{(j-1)} * E_i$ which by (10) and (11) is the same as the Gram matrix for $(\Psi_x^{(j)} : x \in S)$. Thus, by proposition 2 for each $j \in \{1, \ldots, t\}$ there is a unitary operator $U^{(j)}$ satisfying $\Psi_x^{(j)} = U^{(j)} O \Psi_x^{(j-1)}$ for each $x \in S$.

The sequence $(U^{(j)} : 0 \leq j \leq t)$ and the indexed sets $(\Psi_x^{(j)} : x \in S)$ for $0 \leq j \leq t$ satisfy the conditions (3) and thus they correspond to the execution of the query stage of a QQA. Having defined $(U^{(j)} : 0 \leq j \leq t)$, it remains to show that there is a CSOP $(P_z : z \in T)$ that gives the output stage of the QQA.

By (12), $M^{(t)} = \sum_{z \in T} \Gamma_z$. By the lower bound on $\dim(H_A)$ we can write H_A as a direct sum of orthogonal spaces $\oplus_{z \in T} H_z$ where $\dim(H_z) \geq \text{rank}(\Gamma_z)$ and let $(Q_z : z \in T)$ be the associated CSOP. By lemma 1, there is a unitary matrix U such that Γ_z is the Gram matrix of $(Q_z U \Psi_x^{(t)} : x \in S)$ and by Proposition 1, Γ_z is also the Gram matrix of $(P_z \Psi_x^{(t)} : x \in S)$ where $(P_z : z \in T)$ is the CSOP defined by $P_z = U^{-1} Q U$. ■

4. The final state Gram matrix

The following proposition characterizes the matrices that can arise as final state Gram matrices of a QQA:

Proposition 3 *Let $S \subseteq \{0,1\}^n$ and let M be an $S \times S$ matrix with complex entries. Then the following conditions are equivalent:*

1. *There is a QQA whose final state Gram matrix is M,*
2. *There is a family of unit vectors $(\Phi_x : x \in S)$ in some Hilbert space H whose Gram matrix is M*
3. *M is Hermitian, positive semidefinite, with all diagonal entries equal to 1 and all off-diagonal entries of magnitude at most 1.*

Proof: The implications $(1) \Longrightarrow (2) \Longrightarrow (3)$ are immediate, so it remains to prove $(3) \longrightarrow (1)$. Let M be a matrix satisfying (3).

We need to define the query stage of an algorithm that ends with M as its final state Gram matrix. To do this, let r be the rank of M and let the working memory of the quantum computer have two parts; the first consisting of n single bit registers and the second consisting of one register with value set $\{1, \ldots, r\}$. Let H_{W_1} and H_{W_2} be the corresponding Hilbert spaces, and $H_W = H_{W_1} \otimes H_{W_2}$. The accessible memory is then $H_Q \otimes H_{W_1} \otimes H_{W_2}$ and we write a basis state of H_A as $|q_1\rangle|w_1\rangle|w_2\rangle$.

By Proposition 1 there is an indexed family of vectors $\{\Phi_x : x \in S\} \subseteq H_{W_2}$ whose Gram matrix is M (which must be unit vectors since the diagonal entries of M are all 1).

The algorithm consists of $2n$ queries. The first n queries are used to copy the input onto W_1, to reach the system of state vectors is $(|0\rangle|x\rangle|0\rangle : x \in S)$. Next a unitary operation is applied to bring the system of state vectors to $(|0\rangle|x\rangle\Phi_x : x \in S)$ (here we use the fact that the two systems each consist of pairwise orthogonal vectors so there is a unitary transformation that maps one system to the other). The next n queries are done to reset W_1 to $|0\rangle$ thus leaving the system of state vectors as $(|0\rangle|0\rangle\Psi_x : x \in S)$. This system has the desired Gram matrix. The details of defining the QQA are technical but standard.

■

We refer to any matrix M that satisfies these three equivalent conditions as a *realizable state matrix*.

Focusing on the final state Gram matrix gives an alternative view of the SDP $P(f, t, \varepsilon)$. We think of $M = M^{(t)}$ as a fixed realizable state matrix and split the SDP into two separate SDPs. Let us define:

Semidefinite program $Q(M, t)$. Find $S \times S$ real symmetric positive semidefinite matrices $(M_i^{(j)} : 0 \leq j \leq t, 1 \leq i \leq n)$ satisfying (9), (10) and (11).

Semidefinite program $O(M, f, \varepsilon)$. Find $S \times S$ real symmetric positive semidefinite matrices $(\Gamma_z : z \in T)$ satisfying (12) and (13).

The semidefinite program $Q(M, t)$, the *query SDP* is feasible if and only if M can be realized as the final state matrix of a t-step QQA. Notice that $Q(M, t)$ is independent of the function f and the error parameter ε. The semidefinite program $O(M, f, \varepsilon)$, the *output SDP* is feasible if and

Proceedings of The 18th IEEE Annual Conference on Computational Complexity
ISBN 0-7695-1879-6/03 $17.00 © 2003 IEEE

only if f is computable from M within error ε, which means that for any system $(\Psi_x : x \in S)$ of unit vectors whose Gram matrix is M there is a CSOP $(P_z : z \in T)$ that satisfies the output conditions (4) for computing f within ε.

The SDP $P(f, t, \varepsilon)$ can then be expressed as the problem of finding an M such that both $Q(M, t)$ and $O(M, f, \varepsilon)$ are feasible.

5. The Ambainis output condition and the zero-error case

As discussed in the introduction, the idea of studying the evolution of the Gram matrices associated with the quantum state of a quantum query computation was introduced by Ambainis to prove lower bounds. Ambainis showed that for an $S \times S$ realizable Gram matrix M, a necessary condition for f to be computable by M within error ε is:

Condition $A(M, f, \varepsilon)$: For all $x, y \in \{0,1\}^n$ with $f(x) \neq f(y)$ we have

$$|M^{(t)}[x,y]| \leq 2\sqrt{\varepsilon(1-\varepsilon)} \quad \text{(Ambainis [3, 4])} \tag{15}$$

It is easy to see that for any realizable Gram matrix M, the feasibility of $O(M, f, \epsilon)$ implies $A(M, F, \epsilon)$. Suppose $(\Gamma_z : z \in T)$ is a feasible solution to $O(M, f, \epsilon)$, and let $x, y \in S$ with $f(x) \neq f(y)$. Let $\Gamma' = M - \Gamma_{f(x)}$. As M has 1's on the diagonal and Γ_z and Γ' are each positive semidefinite symmetric,

$$\begin{aligned}
|M_t[x,y]| &\leq |\Gamma_z[x,y]| + |\Gamma'[x,y]| \\
&\leq \sqrt{\Gamma_z[x,x]}\sqrt{\Gamma_z[y,y]} \\
&\quad + \sqrt{\Gamma'[x,x]}\sqrt{\Gamma'[y,y]} \\
&= \sqrt{\Gamma_z[x,x]}\sqrt{1-\Gamma'[y,y]} \\
&\quad + \sqrt{1-\Gamma'[x,x]}\sqrt{\Gamma'[y,y]} \\
&\leq \sqrt{\Gamma_z[x,x] + \Gamma'[y,y]} \\
&\quad \times \sqrt{(1-\Gamma_z[y,y]) + (1-\Gamma'[x,x])} \\
&\leq 2\sqrt{1-\varepsilon}\sqrt{\varepsilon},
\end{aligned}$$

where the next to last inequality follows from the Cauchy-Schwartz inequality and the final inequality comes from (13) which implies $\Gamma_z[x,x] \geq 1-\varepsilon$ and $\Gamma'[y,y] \geq 1-\varepsilon$.

It is natural to ask whether we could simplify $P(f, t, \varepsilon)$ by replacing the output conditions $O(M^{(t)}, f, \varepsilon)$ by the simpler condition $A(M^{(t)}, f, \varepsilon)$ in $P(f, t, \varepsilon)$, thereby eliminating the variables Γ_z. Let us define:

Semidefinite program $P_A(f, t, \varepsilon)$. Find $S \times S$ real positive symmetric semidefinite matrices $(M_i^{(j)} : 0 \leq j \leq t-1, 0 \leq i \leq n)$ and $M^{(t)}$ satisfying $Q(M^{(t)}, t)$ and condition $A(M, f\varepsilon)$.

What we've shown above is that any feasible solution to $P(f, t, \varepsilon)$ gives a feasible solution to $P_A(f, t, \varepsilon)$, and the question is whether the converse holds. In section 10 we will consider this question for $\varepsilon > 0$. Here we note that for $\varepsilon = 0$ the converse holds in a strong sense:

Proposition 4 *Let M be a $S \times S$ realizable state matrix. Then $O(M, f, 0)$ is feasible if and only if M satisfies condition $A(M, f, 0)$.*

Proof: If a realizable state matrix M satisfies condition $A(M, f, 0)$, M must be block diagonal with respect to the partition of S into sets $(f^{-1}(z) : z \in T)$ and we can simply define Γ_z so that it coincides with M on the block $f^{-1}(z)$ and is 0 everywhere else. ∎

6. Estimating the quantum query complexity

In this section, we observe that standard algorithms for semidefinite programming can be applied to estimate the quantum query complexity of a function f in time polynomial in the size of its truth table. Previously, no subexponential algorithm was known.

We assume we are given the function f as input, via its truth table, and a real number $\varepsilon \in (0, 1/2)$. We want to estimate $\mathrm{QQC}_\varepsilon(f)$. We will sketch how to estimate this within a factor of 3. In the discussion below we are not concerned with optimizing the polynomial running time bound.

Our algorithm uses the following basic fact about amplifying algorithms. Let $T = \mathrm{QQC}_\varepsilon(f)$ and let A be a quantum If we run three independent trials and take the majority outcome (or any one of the three different outcomes if there isn't a majority) then we get an algorithm that runs in time $3T$ with error probability at most $3\varepsilon^2 - 2\varepsilon^3$.

Now, for each t between 0 and n we want to determine the minimum value δ_{opt} $P(f, t, \delta_{opt})$ is feasible. This is a semidefinite programming optimization problem. For any $\gamma > 0$, semidefinite programming can be used to determine a δ that satisfies $\delta_{opt} \leq \delta \leq \delta_{opt} + \gamma$. We do this for $\gamma = \varepsilon - 3\varepsilon^2 + 2\varepsilon^3$ (which is positive for $\varepsilon \in (0, 1/2)$). If T^* is the least t for which this value of δ is less than or equal to ε, then we can conclude that $\mathrm{QQC}_\varepsilon(f) \leq T \leq \mathrm{QQC}_{\varepsilon - 3\varepsilon^2 + 2\varepsilon^3}(f) \leq 3\mathrm{QQC}_\varepsilon(f)$.

Now the solution of the above approximate SDP optimization problem can be done via the ellipsoid method for general convex programming. The polynomial time algorithms for SDP (see, e.g., [8]) involve various technicalities, including input assumptions and qualifications on the output. We defer discussions to the full paper. For now we mention that the key issue involves finding an initial feasible point that is suitably far from the boundary of the feasible region, and in our case this can be done by taking the feasible solution corresponding to the algorithm that at each

Proceedings of The 18th IEEE Annual Conference on Computational Complexity
ISBN 0-7695-1879-6/03 $17.00 © 2003 IEEE

step performs the null query with amplitude $1/\sqrt{2}$ and each other query with amplitude $1/\sqrt{2n}$. The running time of the algorithm is polynomial in the size of the input, $\log R/r$ where r is the distance of the initial point to the boundary of the feasible region and R is an upper bound on the distance of the optimal solution from the origin.

7. Trees and Branching

The notion of classical (deterministic) decision tree is inseparable from the notion of branching. Is there an analogue of branching for the quantum case? One possible answer is that $M^{(j)} = \sum_{i=0}^{n} M_i^{(j)}$ in $P(f,t,\epsilon)$ can be viewed as an expression for quantum branching.

To better understand how classical and quantum decision trees are related, we describe classical (deterministic) decision trees in terms of the matrices of $P(f,t,\epsilon)$. Without loss of generality we can assume that a decision tree of depth t is a complete binary tree of depth t. Each internal node is labeled by the variable that is queried at that node and each leaf is labeled by the value output upon arriving at that node. For each input x, let $v_j(x)$ be the node of the tree to which x arrives after query j. Define the matrices $(M^{(j)} : 0 \le j \le t)$, $(M_i^{(j)} : 0 \le j \le t, 0 \le i \le n)$ and $(\Gamma_z : z \in T)$ by:

$$M^{(j)}[x,y] = \begin{cases} 1 & \text{if } v_j(x) = v_j(y) \\ 0 & \text{if } v_j(x) \neq v_j(y) \end{cases}$$

$$M_i^{(j)}[x,y] = \begin{cases} 1/2 & \text{if } v_j(x) = v_j(y) \text{ and } i = 0 \\ 1/2 & \text{if } v_j(x) = v_j(y) \text{ and this} \\ & \text{node queries variable } i \\ 0 & \text{otherwise} \end{cases}$$

$$\Gamma_z[x,y] = \begin{cases} 1 & \text{if } v_t(x) = v_t(y) \text{ and this node} \\ & \text{outputs } z \\ 0 & \text{otherwise} \end{cases}$$

8. The Dual Problem

In this section we use the duality theory of semidefinite programming to construct another semidefinite program that characterizes quantum query complexity.

There are various forms of the duality theorem of semidefinite programming in the literature, but we formulate and prove one that is especially suited to our situation. We use an extension of Farkas' lemma which is closely related to Lemma 2.3 of [2]. Let S, J, and I be finite sets. Let $A = (A_{\alpha,\beta} : \alpha \in I, \beta \in J)$ and $B = (B_\alpha : \alpha \in I)$, where each $A_{\alpha,\beta}$ and B_α is a real symmetric $S \times S$ matrix. We define the two semidefinite programs.

Semidefinite program $\Pi(A,B)$. Find a J-tuple $(X_\beta : \beta \in J)$ of real symmetric positive semidefinite $S \times S$ matrices satisfying:

$\sum_{\beta \in J} A_{\alpha,\beta} * X_\beta = B$, for all $\alpha \in I$,

Semidefinite program $\hat{\Pi}(A,B)$. Find symmetric $S \times S$ real matrices $(Y_\alpha : \alpha \in I)$ such that:

$\hat{\Pi}1$. $\sum_{\alpha \in I} Y_\alpha * A_{\alpha,\beta} \preceq \mathbf{0}$,

$\hat{\Pi}2$. The sum of entries of the matrix $\sum_{\alpha \in I} B_\alpha * Y_\alpha$ is positive.

We have:

Lemma 2 *Let S, J, I, A, B be as above, and suppose that the only feasible solution to $\hat{\Pi}(A,\mathbf{0})$ is $X = \mathbf{0}$. Then exactly one of $\Pi(A,B)$ and $\hat{\Pi}(A,B)$ is feasible.*

Proof: First suppose that $(X_\beta : \beta \in J)$ is feasible for Π. Let $(Y_\alpha : \alpha \in I)$ satisfy $\hat{\Pi}1$; we claim that $\hat{\Pi}2$ is violated. Letting $W = \sum_{\alpha \in I} Y_\alpha * B_\alpha$ we have:

$$\begin{aligned} W &= \sum_{\alpha \in I, \beta \in J} Y_\alpha * A_{\alpha,\beta} * X_\beta \\ &= \sum_{\beta \in J} X_\beta * (\sum_{\alpha \in I} A_{\alpha,\beta} * Y_\alpha) \preceq \mathbf{0}, \end{aligned}$$

since the set of positive semidefinite matrices is closed under $*$ and $+$. Therefore, $\hat{\Pi}2$ does not hold since the sum of the entries of W is equal to $\langle e, We \rangle \le 0$ where e is the all 1 vector.

Now suppose that $\Pi(A,B)$ is infeasible. Let P denote the set of all J-tuples $(X_\beta : \beta \in J)$ of $S \times S$ positive semidefinite matrices. Consider the space of all I-tuples $(Z_\alpha : \alpha \in I)$ of $S \times S$ symmetric matrices (which is isomorphic to $\mathbb{R}^{|I||S|(|S|+1)/2}$) and consider the subset K consisting of all tuples of the form $(\sum_{\beta \in J} A_{\alpha,\beta} * X_\beta : \alpha \in I)$ where $(X_\beta : \beta \in J) \in P$. K is a convex cone and is also closed by Theorem 9.1 of [18] and the hypothesis that the only feasible solution to $\hat{\Pi}(A,\mathbf{0})$ is $X = \mathbf{0}$. Since $\Pi(A,B)$ is not feasible, $(B_\alpha : \alpha \in I) \notin K$, and so (see, e.g., Theorem 11.7 in [18]) there is a separating hyperplane, which in this case is a I-tuple $(Y_\alpha : \alpha \in I)$ such that the sum of the entries of $\sum_{\alpha \in I} B_\alpha * Y_\alpha$ is positive, and such that the entry sum of $\sum_{\alpha \in I} Z_\alpha * Y_\alpha$ is nonpositive for all $(Z_\alpha : \alpha \in I) \in K$. This latter condition is equivalent to the condition that for all $(X_\beta : \beta \in J) \in P$, $\sum_{\beta \in J} X_\beta * (\sum_{\alpha \in I} A_{\alpha,\beta} * Y_\alpha)$ has nonpositive entry sum, which is equivalent to the condition that each of the matrices $(\sum_{\alpha \in I} A_{\alpha,\beta} * Y_\alpha : \beta \in J)$ is negative semidefinite. Thus $(Y_\alpha : \alpha \in I)$ is feasible for $\hat{\Pi}$. ■

Proceedings of The 18th IEEE Annual Conference on Computational Complexity
ISBN 0-7695-1879-6/03 $17.00 © 2003 IEEE

To formulate the dual $\hat{P}(f,t,\varepsilon)$ to the SDP $P(f,t,\varepsilon)$ we express $P(f,t,\varepsilon)$ in the form $\Pi(A,B)$ and use the lemma. The form $\Pi(A,B)$ requires that all linear constraints be equalities. Thus we must convert (13) to an equality constraint. This we do by introducing slack variable matrices $(\Omega_z : z \in T)$ that are constrained to be positive semidefinite and rewrite (13) as:

$$\Delta_z * \Gamma_z - \Delta_z * \Omega_z = (1-\epsilon)\Delta_z$$

We also eliminate constraint (11) and the variable matrix $M^{(t)}$ by combining (11) and (12) into the single constraint:

$$\sum_{i=0}^{n} E_i * M_i^{(t-1)} = \sum_{z \in T} \Gamma_z.$$

Now we define the index sets I and J and the systems of matrices A and B corresponding to this modified form of $P(f,t,\epsilon)$. We take $I = \{0,\ldots,t\} \cup T$ and and $J = \{0,\ldots,t-1\} \times \{0,\ldots,n\} \cup T \times \{1,-1\}$. We specify the matrices $A_{\alpha,\beta}$ and B_α as follows (using the notation for matrices introduced prior to the definition of $P(f,t,\varepsilon)$ in section 3).

For $\alpha \in \{0,\ldots,t\}$, $\alpha' \in T$, $\beta = (\beta_1,\beta_2) \in \{0,\ldots,n\} \times \{0,\ldots,r\}$ and $\beta' = (\beta'_1,\beta'_2) \in T \times \{1,-1\}$ we define:

$$\begin{aligned}
A_{\alpha,(\beta_1,\beta_2)} &= \begin{cases} E & \text{if } \beta_1 = \alpha \\ -E_{\beta_2} & \text{if } \beta_1 = \alpha - 1 \\ \mathbf{0} & \text{otherwise} \end{cases} \\
A_{\alpha,(\beta'_1,\beta'_2)} &= \begin{cases} E & \text{if } \alpha = t \text{ and } \beta'_1 = 1 \\ \mathbf{0} & \text{otherwise} \end{cases} \\
A_{\alpha',(\beta_1,\beta_2)} &= \mathbf{0} \\
A_{\alpha',(\beta'_1,\beta'_2)} &= \begin{cases} \beta'_2 \Delta_{\alpha'} & \text{if } \alpha' = \beta'_1 \\ \mathbf{0} & \text{if } \alpha' \neq \beta'_1 \end{cases} \\
B_\alpha &= \begin{cases} E & \text{if } \alpha = 0 \\ \mathbf{0} & \text{if } \alpha > 0 \end{cases} \\
B_{\alpha'} &= (1-\varepsilon)\Delta_1
\end{aligned}$$

It is routine (if tedious) to check that with these definitions $\Pi(A,B)$ is (the modified form of) $P(f,t,\varepsilon)$. Figure 8 provides a visual aid for the verification. We now form $\hat{\Pi}(A,B)$ to form the dual $\hat{P}(f,t,\varepsilon)$ to $P(f,t,\varepsilon)$. We rename the dual variable matrices $(Y_j : j \in \{0,\ldots,t\})$ as $(-L^{(j)} : j \in \{0,\ldots,t\})$ and the dual variables $(Y_z : z \in T)$ as $(\Lambda_z : z \in T)$.

Semidefinite program $\hat{P}(f,t,\varepsilon)$. Find $S \times S$ real symmetric matrices $(L_i : 0 \le i \le t)$ and $(\Lambda_z : z \in T)$ satisfying:

$$\begin{aligned}
(16) \quad E_i * L^{(j)} &\preceq L^{(j-1)} \qquad 1 \le j \le t, 0 \le i \le n. \\
(17) \quad \Delta_z * \Lambda_z &\preceq L^{(t)} \qquad z \in T. \\
(18) \quad \Lambda_z &\succeq \mathbf{0} \qquad z \in T. \\
(19) \quad \sum_{x,y \in S} L^{(0)}[x,y] &< (1-\varepsilon)\mathbf{Tr}(\sum_{z \in T} \Delta_z * \Lambda_z).
\end{aligned}$$

Combining Theorems 1 and Lemma 2 we have:

Theorem 2 *For any partial boolean function f, natural number t and $\varepsilon \in (0,1/2)$ the following are equivalent:*

1. *There is a t-step QQA that computes f within error ε.*
2. *$P(f,t,\varepsilon)$ is feasible.*
3. *$\hat{P}(f,t,\varepsilon)$ is not feasible.*

Proof: The only thing we need is to show that the hypothesis of Lemma 2 holds for the system A of matrices arising from $P(f,t,\epsilon)$. Thus we consider the homogeneous system associated with the primal, obtained by setting the matrices corresponding to $(B_\alpha : \alpha \in I)$ to $\mathbf{0}$. If we do this, then the hypothesis follows by repeatedly applying the fact that if the sum of a list of positive semidefinite matrices is $\mathbf{0}$ then all of the summands are $\mathbf{0}$. ■

The contrapositive of the implication (1)$\Longrightarrow$ (3) is of particular interest: it says that a feasible solution to the dual $\hat{P}(f,t,\varepsilon)$ proves the nonexistence of t-step ε-error QQA for f, in other words it proves a lower of $t+1$ on the quantum query complexity $\mathrm{QQC}_\varepsilon(f)$. We will have more to say about this in the next section.

9. Ambainis' lower bound method revisited

9.1. The dual of $P_A(f,t,\epsilon)$

As discussed in section 5, the SDP $P_A(f,t,\varepsilon)$ is a relaxation of $P(f,t,\varepsilon)$ and therefore, in general, its feasibility is only a necessary condition for the existence of a t-step ε QQA for f. However, this one way implication is still enough for us to conclude that the feasibility of the SDP $\hat{P}_A(f,t,\varepsilon)$ implies that $\mathrm{QQC}_\varepsilon(f) > t$. It is therefore potentially of interest to construct $\hat{P}_A(f,t,\varepsilon)$, which is simpler than $\hat{P}(f,t,\varepsilon)$. Using this dual LP we will derive a simply stated lower bound criterion for quantum query complexity. This lower bound condition encompasses the lower bound method of Ambainis [3, 4] and its generalizations by other researchers, e.g., [5].

Proceedings of The 18th IEEE Annual Conference on Computational Complexity
ISBN 0-7695-1879-6/03 $17.00 © 2003 IEEE

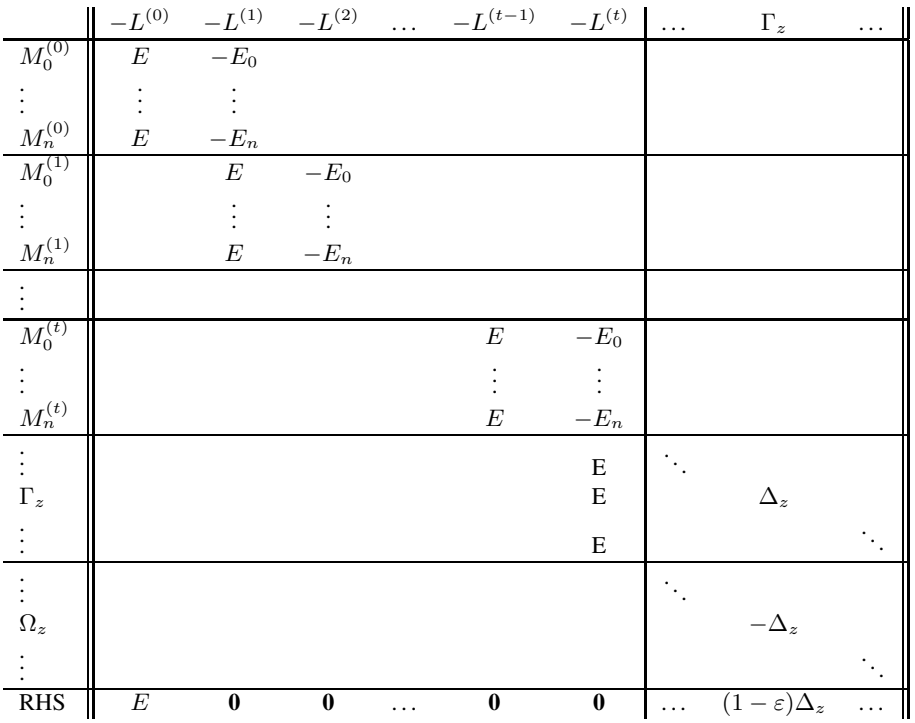

Figure 1. Each entry in the table represents a $S \times S$ matrix. The columns of the table correspond to the set I and are labeled by the dual variable matrices. The rows of the table correspond to the set J and are labeled by the primal variable matrices. Each column gives a constraint of (the modified version of) $P(f,t,\epsilon)$ and each row gives a constraint of $\hat{P}(f,t,\epsilon)$. (This is the transpose of the standard picture, which we use for typographical convenience.) The final row of the table gives B_α for $\alpha \in I$ followed by $B^+_{\alpha'}$ for $\alpha' \in I$. The entries of the upper main part of the table give the matrices $A^0_{\alpha,\beta}$ and of the lower part give $A^0_{\alpha',\beta}$.

Proceedings of The 18th IEEE Annual Conference on Computational Complexity
ISBN 0-7695-1879-6/03 $17.00 © 2003 IEEE

Semidefinite program $\hat{P}_A(f,t,\varepsilon)$ Find $S \times S$ real symmetric matrices $(K^{(j)} : 0 \le j \le t)$ satisfying:

$$E_i * K^{(j-1)} \succeq K^{(j)} \qquad 1 \le j \le t, 0 \le i \le n. \tag{20}$$

$$K^{(0)}[x,y] = 0, \qquad \text{for } f(x) = f(y) \tag{21}$$

$$\sum_{x,y \in S} K^{(t)}[x,y] > 2\sqrt{\epsilon(1-\epsilon)} \sum |K^{(0)}[x,y]|. \tag{22}$$

Theorem 3 *$P_A(f,t,\varepsilon)$ is feasible if and only if $\hat{P}_A(f,t,\varepsilon)$ is infeasible.*

Proof: As we did to form a dual for $P(f,t,\varepsilon)$ we represent $P_A(f,t,\varepsilon)$ in the form of $\Pi(A,B)$ and apply lemma 2. To this end we reexpress $P_A(f,t,\varepsilon)$ in matrix form in terms of $(M_i^{(j)} : 0 \le j \le t-1, 0 \le i \le n\}$. As we did for $P(f,t,\varepsilon)$ we eliminate the primal variables $M^{(t)}$ and combine (11) with the output conditions which, in this case is $A(M^{(t)}, f, \varepsilon)$ to get the two conditions:

$$F * \sum_{i=0}^{n} E_i * M_i^{(t-1)} \le 2\sqrt{\varepsilon(1-\varepsilon)}F.$$

$$-F * \sum_{i=0}^{n} E_i * M_i^{(t-1)} \le 2\sqrt{\varepsilon(1-\varepsilon)}F,$$

where recall that $E_i[x,y] = (-1)^{x_i + y_i}$ and $F[x,y] = 1$ if and only if $f(x) \neq f(y)$.

The representation of $P_A(f,t,\varepsilon)$ in the form $\Pi(A,B)$ is similar to (but simpler than) that of $P(f,t,\varepsilon)$. The index set I is $\{0,\ldots,t-1\} \cup \{+,-\}$. where $\{+,-\}$ are the indices assigned to the two new output constraints above. The variables $(\Gamma_z : z \in T)$ are gone so T is removed from the index set J leaving just $\{0,\ldots,t-1\} \times \{0,\ldots,n\} \cup \{+,-\}$ where $\{+,-\}$ are the indices for two slack variable matrices Ω_+ and Ω_-. For $\alpha \in \{0,\ldots,t-1\}$, $\alpha' \in \{+,-\}$, $(j,i) \in \{0,\ldots,t-1\} \times \{0,\ldots,n\}$ and $\beta \in \{+,-\}$ we define $A_{\alpha,(j,i)}$ and B_α as for $P(f,t,\varepsilon)$. We define $A_{\alpha,\beta'} = \mathbf{0}$. $A_{\alpha',(j,i)}$ is $\mathbf{0}$ for $j < t-1$ and is $\alpha' F * E_i$ for $j = t-1$, and $B_{\alpha'}$ is $2\sqrt{\varepsilon(1-\varepsilon)}F$. Finally $A_{\alpha',\beta'} = F$ if $\alpha' = \beta'$ and is $\mathbf{0}$ otherwise. Finally $B_{\alpha'} = 2\sqrt{\epsilon(1-\epsilon)}F$.

When we apply Lemma 2 to this, the dual variables $L^{(0)},\ldots,L^{(t-1)}$ are the same as when we formed the dual of $P(f,t,\varepsilon)$ and they satisfy (16) for $j \le t-1$. The dual variables $(\Lambda_z : z \in T)$ are eliminated and $L^{(t)}$ is replaced by two matrices $L_1^{(t)}$ and $L_2^{(t)}$. We have the additional constraints:

$$L^{(t-1)} \preceq F * E_i * (L_2^{(t)} - L_1^{(t)}) \qquad 0 \le i \le n, \tag{23}$$

$$F * L_h^{(t)} \succeq \mathbf{0} \qquad h \in \{1,2\}. \tag{24}$$

$$\sum_{x,y \in S} L^{(0)}[x,y] < 2\sqrt{\varepsilon(1-\varepsilon)} \times \sum_{x,y \in S} F * (L_1^{(t)} + L_2^{(t)})[x,y]. \tag{25}$$

We can reexpress the above system by replacing $L_1^{(t)}, L_2^{(t)}$ by the pair $L^{(t)}, J^{(t)}$ of matrices defined by $L^{(t)} = L_2^{(t)} - L_1^{(t)}$ and $J^{(t)}[x,y] = \min\{L_1^{(t)}[x,y], L_2^{(t)}[x,y]\}$. Then $L_2^{(t)} - L_1^{(t)}$ is replaced by $L^{(t)}$ in (23), constraint (24) is replaced by $J^{(t)} \ge 0$ and $L_1^{(t)} + L_2^{(t)}[x,y]$ is replaced by $|L^{(t)}[x,y]| + 2J^{(t)}[x,y]$ in (25). We then note that since $J^{(t)}[x,y] \ge 0$, (25), $J^{(t)}$ can be set to $\mathbf{0}$ in any feasible solution, and thus we can eliminate $J^{(t)}$ from (25) and remove (24). Next, we note that the entries $L^{(t)}[x,y]$ for $f(x) = f(y)$ are irrelevant and so we may impose the condition $L^{(t)}[x,y] = 0$ for $f(x) = f(y)$. This allows us to eliminate F from the (23) and (25).

Finally we replace $L^{(j)}$ by $-K^{(t-j)}$ for $0 \le j \le t$ to get $\hat{P}_A(f,t,\varepsilon)$. ■

9.2. A general form for Ambainis' lower bound

Using Theorem 3, we now obtain an elegant general form of Ambainis' lower bound method. For a matrix M, let $\lambda(M)$ denote the greatest eigenvalue of M.

Theorem 4 *Let $f : S \longrightarrow T$ be a partial boolean function with $S \subseteq \{0,1\}^n$. Let Γ be an arbitrary $S \times S$ nonnegative symmetric matrix that satisfies $\Gamma[x,y] = 0$ whenever $f(x) = f(y)$. For $i \in \{1,\ldots,n\}$ let Γ_i be the matrix:*

$$\Gamma_i[x,y] = \begin{cases} 0 & \text{if } x_i = y_i \\ \Gamma[x,y] & \text{if } x_i \neq y_i. \end{cases}$$

Then:

$$QQC_\varepsilon(f) \ge \frac{(1 - 2\sqrt{\epsilon(1-\epsilon)}\lambda(\Gamma)}{2\max_{1 \le i \le n} \lambda(\Gamma_i)}.$$

Proof: We show that given Γ as hypothesized and an integer t that is less than $(1 - 2\sqrt{\epsilon(1-\epsilon)})\lambda(\Gamma)/(2\max_{1 \le i \le n} \lambda(\Gamma_i))$, we can construct a feasible solution $(K^{(j)} : 0 \le j \le t\}$ to $\hat{P}(f,t,\varepsilon)$. Since Γ and Γ_i are nonnegative matrices, they have nonnegative largest eigenvalue. Furthermore, Γ has a unit eigenvector v with nonnegative entries (see, e.g., [14]). Let D be the diagonal matrix with $D[x,x] = (v_x)^2$. We choose $K^{(j)} = (\Gamma - j\alpha I) * vv^T$ where $\alpha = 2\max_{1 \le i \le n} \lambda(\Gamma_i))$.

Proceedings of The 18th IEEE Annual Conference on Computational Complexity
ISBN 0-7695-1879-6/03 $17.00 © 2003 IEEE

We now show that $(K^{(j)} : 0 \leq j \leq t)$ is feasible for $\hat{P}_A(f,t,\varepsilon)$. Condition (21) is immediate. For condition (20) we have for $1 \leq j \leq t$ and $0 \leq i \leq n$:

$$\begin{aligned} E_i * K^{(j-1)} - K^{(j)} &= ((\Gamma - (j-1)\alpha I) * E_i \\ &\quad -(\Gamma - j\alpha I)) * vv^T \\ &= (\alpha I - 2\Gamma_i) * vv^T. \end{aligned}$$

This matrix is positive semidefinite since, by the definition of α, $\alpha I - 2\Gamma_i$ is.

Finally, (22) holds since the sum of the entries of $K^{(0)}$ is equal to $v^T \Gamma v = \lambda(\Gamma)$, while the sum of the entries of $K^{(t)}$ is $\lambda(\Gamma) - t\alpha \geq 2\sqrt{\epsilon(1-\epsilon)}\lambda(\Gamma)$. ■

Let us relate Theorem 4 to Ambainis' lower bound method. Ambainis main lower bound criterion (Theorem 5.1 in [4]) is obtained by taking Γ to be a 0-1 matrix (i.e. the characteristic matrix of a binary relation on S), and using the facts that $\lambda(\Gamma) \geq x^T \Gamma x$ for any unit vector x and that $\lambda(\Gamma_i)$ is bounded above by the maximum row sum (since Γ_i is nonnegative).

The generalization of Ambainis' method by Barnum and Saks (Theorem 1 in [5]) gives a lower bound of a similar form to Theorem 4 but the denominator is the maximum of $\beta(\Gamma_i)$ where $\beta(M)$ is the maximum over x, y such that $M[x,y] \neq 0$ of $\sqrt{(\sum_z M_{x,z})(\sum_z M_{x,y})}$. It is not hard to show that $\beta(M) \geq \lambda(M)$ for any nonnegative matrix.

9.3. The zero error case

In the case of zero error, $\hat{P}_A(f,t,\varepsilon)$ simplifies even further:

Semidefinite program $\hat{P}_A(f,t,0)$ Find $S \times S$ real symmetric matrices $(K^{(j)} : 0 \leq j \leq t)$ satisfying:

$$\begin{aligned} &(26) \quad E_i * K^{(j-1)} \succeq K^{(j)} \qquad 1 \leq j \leq t, 0 \leq i \leq n. \\ &(27) \quad K^{(0)}[x,y] = 0, \qquad \text{for } f(x) = f(y) \\ &(28) \quad \sum_{x,y \in S} K^{(t)}[x,y] > 0. \end{aligned}$$

By Proposition 4, $P(f,t,0)$ and $P_A(f,t,0)$ are either both feasible or both infeasible. Therefore:

Theorem 5 *For any partial boolean function f and natural number t the following are equivalent:*

1. *There is a t-step QQA that computes f with 0 error.*
2. *$P_A(f,t,0)$ is feasible.*
3. *$\hat{P}_A(f,t,0)$ is not feasible.*

Example: Consider $f = x_1 \wedge x_2$. We show that the (zero-error) quantum decision tree complexity of this function is greater than 1 (i.e. it is 2) by giving $L_{(0)}$ and $L_{(1)}$ that satisfy the conditions of $\hat{P}_A(f,t,0)$:

$$L^{(1)} = \begin{pmatrix} 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 2 \\ -2 & 2 & 2 & 0 \end{pmatrix};$$

$$L^{(0)} = \begin{pmatrix} -6 & 0 & 0 & 5 \\ 0 & -6 & 0 & 5 \\ 0 & 0 & -6 & 5 \\ 5 & 5 & 5 & -11.5 \end{pmatrix}$$

Here the rows and columns of the matrices correspond to inputs 00, 01, 10, 11 in this order.

10. How strong is the Ambainis output condition?

In Section 5 we introduced the Ambainis output condition and used it to formulate a semidefinite program $P_A(f,t,\varepsilon)$ that is simpler than $P(f,t,\varepsilon)$. We noted that the feasibility of $P(f,t,\varepsilon)$ implies the feasibility $P_A(f,t,\varepsilon)$ and that for $\varepsilon = 0$ the implication goes both ways. We strongly suspect that this is not true for general ε but do not have a proof of this.

The equivalence of the SDP's when $\varepsilon = 0$ gave the stronger fact that for any realizable state matrix M that satisfies $A(M,f,0)$, $O(M,f,0)$ is feasible. Here we show this stronger result does not extend to $\varepsilon > 0$.

Theorem 6 *For $\epsilon \in (0,1/2)$ and sufficiently large n and $S = \{0,1\}^n$ there is a function $f : S \to \{0,1\}$ and an $S \times S$ realizable state matrix M that satisfies $A(M,f,\varepsilon)$, but where $O(M,f,\varepsilon)$ is not feasible.*

Proof of Theorem 6: Let $m = n^\alpha$, where $\alpha > 1$ is a sufficiently large constant to be determined later. We first observe that there is a system $(\Psi_x : x \in S)$ of unit vectors in H satisfying:

$$(29) \quad \langle \Psi_x \mid \Psi_y \rangle \leq 2\sqrt{\epsilon(1-\epsilon)} \qquad \text{for } x \neq y \in S.$$

It suffices to show that if we choose the vectors independently at random from H (using the standard Lebesgue measure), then (29) holds with positive probability. Let p be the probability that random vectors that $\langle \Psi_x \mid \Psi_y \rangle > 2\sqrt{\epsilon(1-\epsilon)}$ for a particular x, y; it is enough that $p < 2^{-2n}$, which is indeed the case if α is a large enough constant.

So select a system $(\Psi_x : x \in S)$ of unit vectors satisfying (29). Then the associated Gram matrix M satisfies $A(M,f,\varepsilon)$ for *every* $f : \{0,1\}^n \to \{0,1\}$.

Proceedings of The 18th IEEE Annual Conference on Computational Complexity
ISBN 0-7695-1879-6/03 $17.00 © 2003 IEEE

We will now show by a counting argument that there is an f for which $O(M, f, \varepsilon)$ does not hold. Let $H = H_0 \oplus H_1$ be a decomposition into orthogonal subspaces, each of dimension at least 2^n and let (P_0, P_1) be the associated CSOP. If $f : \{0,1\}^n \longrightarrow \{0,1\}$ is a boolean function and Γ_0, Γ_1 are matrices that satisfy $O(M, f, \varepsilon)$ then Lemma 1 implies that there is a unitary matrix U_f such that for $z \in \{0,1\}$, $(P_z U_f \Psi_x : x \in S)$ has Gram matrix Γ_z. Suppose that such U_f exists for every boolean function; we derive a contradiction. Let $\tilde{U}_f$ be the matrix obtained by rounding down the real and imaginary parts of each entry of U_f to the nearest multiple of 2^{-N}, where $N = m + \lambda$ where λ is an integer chosen so that $\sqrt{1-2\epsilon} < 2^{2-\lambda}$. Since U_f are unitary $m \times m$ matrices, and each entry is a complex number of modulus at most 1 and thus the number of distinct matrices of the form $\tilde{U}_f$ is at most 2^{2Nm^2}. Since there are 2^{2^n} boolean functions, there are two functions f and g for which $\tilde{U}_f = \tilde{U}_g$. Let x be an input with $f(x) \neq g(x)$ and without loss of generality assume $f(x) = 1$. The vector $P_1 U_f \Psi_x$ must satisfy $\|P_1 U_f \Psi_x\|^2 = 1 - \epsilon$ while the vector $P_1 U_g \Psi_x$ must satisfy $\|P_1 U_g \Psi_x\|^2 \leq \epsilon$. But

$$\begin{aligned} \frac{1}{2}\sqrt{1-2\epsilon} &\leq \sqrt{1-\epsilon} - \sqrt{\epsilon} \\ &\leq \|P_1 U_f \Psi_x\| - \|P_1 U_g \Psi_x\| \\ &\leq i \quad \|P_1 (U_g - U_f)\Psi_x\| \\ \leq \|(U_g - U_f)\Psi_x\|. \end{aligned}$$

Since Ψ_x is a unit vector, this latter quantity is at most 2^m times the magnitude of the largest entry in $U_g - U_f$. Since this magnitude is at most $2^{1-N} < 2^{-m-1}\sqrt{1-2\epsilon}$, we obtain the desired contradiction.

■

Acknowledgments

We thank Andris Ambainis and Yaoyun Shi for helpful comments and discussions.

References

[1] S. Aaronson, "Algorithms for Boolean function query properties," to appear in SIAM Journal on Computing.

[2] F. Alizadeh, "Interior point methods in semidefinite programming with applications to combinatorial optimization," SIAM J. Optimization, 5 (1995), pp. 13–51.

[3] 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.

[4] A. Ambainis, "Quantum lower bounds by quantum arguments," Journal of Computer and System Sciences, 64(2002), pp. 750-767.

[5] H. Barnum, M. E. Saks "A lower bound on the quantum query complexity of read-once functions," Electronic Colloquium on Computational Complexity (ECCC)(002): (2002)

[6] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf, "Quantum lower bounds by polynomials," Proc. 39th IEEE Symp. on Foundations of Comp. Sci.(FOCS), 1998, pp. 352–361.

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

[8] M. Grötschel, L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization* Algorithms and Combinatorics 2, Springer-Verlag, Berlin, 1988.

[9] Lov Grover, "A fast quantum mechanical algorithm for database search," *Proc. 28th ACM Symp. on Theory of Computing (STOC)*, 1998, pp. 212–219.

[10] L. Grover, "Quantum mechanics helps in searching for a needle in a haystack," *Physical Review Letters*, 79(1997) pp. 325–328.

[11] P. Hoyer, J. Neerbek, and Y.Shi, "Quantum bounds for ordered searching and sorting," Proc. 28th Int. Coll. on Automata Lang. and Prog. (ICALP), 2001, Lecture Notes in Computer Science, vol. 2076, Springer-Verlag, New York, pp. 346–357.

[12] P. Hoyer, J. Neerbek, Y. Shi, "Quantum Complexities of Ordered Searching, Sorting, and Element Distinctness" Algorithmica 34 (2002) pp. 429–448.

[13] A. Kitaev "Any quantum coin tossing protocol is subject to same-sided bias", 2002, unpublished.

[14] H. Minc, *Nonnegative Matrices*, Wiley-Interscience, New York, 1988.

[15] A. Nayak, personal communication.

[16] M.A. Nielsen and I.L. Chuang, *Quantum Computation and Quantum Information* Cambridge University Press, Cambridge, 2000.

[17] A. Peres, *Quantum Theory: Concepts and Methods*, Kluwer Academic, Dordrecht, 1993.

[18] R.T. Rockafeller, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.

[19] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," *Proc. 37th Ann. Symp. on the Foundations of Comp. Sci. (FOCS)*, 1994, pp. 56–65.

[20] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comp.* 26 (1997) pp. 1484–1509.

Proceedings of The 18th IEEE Annual Conference on Computational Complexity
ISBN 0-7695-1879-6/03 $17.00 © 2003 IEEE