Proceedings of Symposia in Applied Mathematics

A Survey of Quantum Complexity Theory

Umesh V. Vazirani

Abstract. This paper provides an introduction to quantum complexity theory. Two basic models of quantum computers, quantum Turing Machines and quantum circuits, are defined and shown to be polynomially equivalent. The quantum complexity class BQP of is defined, and its relationship to classical complexity classes is established. The basic techniques for proving limits on quantum computers, by showing a lower bound in the black box model are surveyed. Finally the quantum analog of the Cook-Levin theorem is sketched.

Contents

1. Introduction

The main goal of computational complexity theory is to classify computational problems according to the amount of computational resources — typically time or number of steps — required to solve them on a computer. That this quantity is well defined rests upon the modern form of the Church-Turing thesis, which asserts that any "reasonable" model of computation can be *efficiently* simulated on a probabilistic Turing Machine. (An efficient simulation is one whose running time is bounded by some polynomial in the running time of the simulated machine.)

2000 *Mathematics Subject Classification.* [2000]Primary 81P68; Secondary 81-01.

Key words and phrases. Quantum complexity theory, quantum Turing machines, lower bounds, NP, quantum computation.

This work was partially supported by DARPA Grant F30602-00-2-0601 and NSF Grant NSF Grant CCR-9800024. (Copyright 2000.)

This thesis may be informally summarized as follows: All physical implementations of computing devices can be simulated with polynomial factor overhead in running time by the probabilistic Turing Machine or the random access machine. Recently, this paradigm has been fundamentally challenged, since there are strong arguments showing that the Church-Turing thesis (in this modern form) does not hold at the level of quantum mechanics.

Early indications of this possibility occurred in the paper by Feynman [**14**] which pointed out that it was not clear how to simulate quantum mechanical systems of n particles (say n spins) on a computer, without paying an exponential penalty in simulation time. The first formal evidence that quantum computers violate the modified Church-Turing thesis came a decade later, with the result of Bernstein and Vazirani [**9**], which showed that relative to an oracle, quantum polynomial time properly contains probabilistic polynomial time. This built upon a previous algorithm due to Deutsch and Jozsa [**13**]. Simon [**24**] improved this bound by showing that relative to an oracle, quantum polynomial time is not contained in subexponential probabilistic time. Shor [**25**] followed up with seminal results in quantum algorithms, showing that the problems of prime factorization and discrete logarithms can both be solved in polynomial time on a quantum computer. The computational intractability of these problems for classical computers is the standard computational assumption underlying modern cryptography. Together, these results provide very strong evidence that quantum computers violate the modern Church-Turing thesis.

In view of these developments, one must explore a new complexity theory based on quantum mechanics. The formal models for quantum computers - quantum Turing Machines and quantum circuits - were introduced by Deutsch [**11**]. These two models were shown to be polynomially equivalent in terms of their computing power by Yao [**31**]. Creating a universal quantum Turing Machine (a programmable quantum Turing Machine) has proved to be a much more challenging task than the corresponding classical universal constructions. The first part of this paper introduces these models, and sketches the universal quantum Turing Machince construction from [**9**], and the polynomial equivalence of quantum Turing Machines and quantum circuits from [**31**]. It also describes universal families of quantum circuits [**7**].

The class of problems that can be solved in polynomial time (in the length of the input) has a special place in complexity theory — since it is regarded as the class of efficiently solvable problems. The class of problems that can be solved in polynomial time on a probabilistic Turing Machine is denoted by BPP. The corresponding class problems solvable in polynomial time on a quantum Turing Machine is denoted by BQP. The best upper bound known for this class is $BQP \subseteq P^{\#P} \subseteq PSPACE$ [**9**], i.e., every problem that can be solved in polynomial time on a quantum Turing Machine can also be solved using polynomial amount of memory space on a classical Turing Machine. Thus $P \subseteq BPP \subseteq BQP \subseteq P^{\#P} \subseteq PSPACE$. Since $P =?PSPACE$ is a major open question in computational complexity theory, this implies that any absolute results showing that quantum computers are more powerful than classical computers ($BQP \neq BPP$) will have to await a major breakthrough in complexity theory. Until then, we must be satisfied with evidence such as [**9, 24, 25**] that quantum computers violate the modern Church-Turing thesis.

Is $NP \subseteq BQP$? In view of the exponential speedups offered by quantum computers for certain computational problems, it is natural to ask whether quantum

computers can solve NP-complete problems in polynomial time. An affirmative answer would have astounding consequences, since the NP-complete problems include several thousand of the most important computational problems, and are believed to be classically intractable. It was shown in [**10**] that relative to a random oracle, a quantum Turing Machine must take exponential time to solve NP-complete problems. This appears to rule out any efficient quantum algorithm for an NP-complete problem, barring a major breakthrough in computational complexity theory. In the same paper an exponential lower bound was established on the problem of inverting a random permutation (thus opening up the possibility of quantum one-way functions). Both results were proved using the hybrid argument. Two other techniques for establishing lower bounds have been introduced. The first is the method of polynomials [**8**], which was used to give a tight linear lower bound on the quantum complexity of the parity function in the black box model. In the same paper they showed that, in general, the quantum query complexity of any total function in the black box model is bounded by the sixth power of the deterministic query complexity. The second technique is the method of quantum adversaries [**2**]. This technique appears to be very general, and has been used to obtain tight bounds for a number of problems. In particular, it was used to prove a tight bound on the problem of inverting a random permutation.

Does $BQP \subseteq NP$? Since BQP includes the ability to randomize, the fair way of asking this question is whether BQP is contained in MA — the probabilistic generalization of NP. There are indications that the answer is negative, since the recursive Fourier sampling problem, which has an efficient quantum algorithm [**9**] is not in MA relative to an oracle. A major question that remains open is whether $BQP \in BPP^{NP}$, i.e., is the ability to do approximate counting sufficient to simulate BQP in polynomial time?

Perhaps the center piece of classical complexity theory is the Cook-Levin theorem, which states that $3-SAT$ is NP-complete. Recently, Kitaev [**18**] proved the quantum analogue of this result. He showed that the problem of 'local Hamiltonians', which is a natural generalization of $3-SAT$ is complete for BQNP. A very non-trivial consequence that follows from this result is that $BQNP \subseteq P^{\#P}$. Our exposition of these results is based on the manuscript [**1**]. We do not know of any natural examples, other than 'local Hamiltonians', of complete problems for this quantum analog of NP. Developing this theory further is an important open question in quantum complexity theory.

A number of beautiful and deep results in classical complexity theory have emerged out of a study of interactive proof systems (see [**23**]). Recently two striking results have been proved about QIP, the quantum analog of IP (interactive polynomial time). The first of these shows that for quantum interactive proofs, three rounds are as powerful as polynomially many rounds [**28, 19**]. The second gives a non-trivial upper bound, showing that $QIP \subseteq EXP$ [**19**].

In addition to time or number of steps, another complexity measure of interest is space, or number of tape cells used by the computation. In this measure, quantum computation offers at most a polynomial factor advantage over classical computation, since it has been shown that a theorem analogous to Savitch's theorem holds for quantum computation, i.e., $QSPACE(f(n)) \subseteq SPACE(f(n)^2)$ [**29**].

This survey is based on lecture notes from a course on quantum computation that I taught recently. The lecture notes from that course are posted on my web page www.cs.berkeley.edu/∼vazirani.

2. Quantum Turing Machines and Quantum Circuits

Just as a bit (an element of $\{0,1\}$) is a fundamental unit of classical information, a qubit is the fundamental unit of quantum information. A qubit is described by a unit vector in the 2 dimensional Hilbert space $\mathcal{C}^2$. Let $|0\rangle$ and $|1\rangle$ be an orthonormal basis for this space. In general, the state of the qubit is a linear superposition of the form $\alpha|0\rangle + \beta|1\rangle$. The state of n qubits is described by a unit vector in the n-fold tensor product $\mathcal{C}^2 \otimes \mathcal{C}^2 \otimes \cdots \otimes \mathcal{C}^2$. An orthonormal basis for this space is now given by the 2^n vectors $|x\rangle$, where $x \in \{0,1\}^n$. This is often referred to as the computational basis. In general, the state of n qubits is a linear superposition of the 2^n computational basis states. Thus the description of an n qubit system requires 2^n complex numbers. This is the source of the astounding information processing capabilities of quantum computers. An important constraint that quantum mechanics places is that the evolution of the state of the n qubits over time must be unitary. In other words, if $|\phi_0\rangle$ is the initial state of the n qubits, and U is the (2^n dimensional) unitary transformation that describes the evolution of the system, then the state after t steps is $|\phi_t\rangle = U^t|\phi_0\rangle$. For our purposes, it is sufficient to consider the results of measuring the state of a quantum system in the computational basis. If the system is in the superposition $|\Phi\rangle = \sum_x \alpha_x |x\rangle$, then measuring the state of the system yields the outcome $|x\rangle$ with probability $|\alpha_x|^2$. See [**22**] for an excellent reference for quantum computation.

The two main models of quantum computers: quantum Turing Machines and quantum circuits were first defined by Deutsch [**11, 12**]. Quantum Turing Machines are much more complex than their classical counterparts, and the construction of universal quantum Turing Machines, as well as implementing programming primitives such as branching and looping requires a considerable amount of work. We section starts with a sketch of these ideas about quantum Turing Machines. Then we turn our attention to quantum circuits, which provide a convenient language in which to describe quantum algorithms. We end with a sketch of Yao's proof showing the equivalence of these two models.

2.1. Quantum Turing Machines.

The definition of a quantum Turing Machines incorporates these concepts into the usual definition of a classical Turing Machine.

Recall that a classical Turing Machine[1] is defined by a triplet (Σ, Q, δ) where Σ is a finite alphabet with an identified blank symbol $\#$, Q is a finite set of states with an identified initial state q_0 and final state q_f, and δ, the *deterministic finite state control*, is a function

$$\delta \quad : \quad Q \times \Sigma \rightarrow \Sigma \times Q \times D$$

with $D = \{L, R\}$. The Turing Machine has a two-way infinite tape of cells indexed by the integers $\mathbf{Z}$ and a single read/write tape head that steps along the tape.

[1] [**23**] is an excellent reference.

A configuration of the TM consists of the description of the contents of the tape, the location of the tape head, and the state $q \in Q$. It is standard to restrict attention to configurations where all but a finite number of tape cells contain the blank symbol #. Each configuration of the TM has a successor defined as follows: If $\delta(p, \sigma) = \tau, q, d$, then whenever the TM is in state p with symbol σ under the tape head, it replaces the symbol σ with the symbol τ, enters state q, and steps in direction d, either left one cell or right one cell.

The TM *halts* on input x if it eventually enters the final state q_f. The number of steps a TM takes to halt on input x is its *running time* on input x. If a TM halts then its *output*is a string in Σ^* consisting of the tape contents from the leftmost non-blank symbol to the rightmost non-blank symbol, or the empty string if the entire tape is blank. A TM which halts on all inputs therefore computes a function from $(\Sigma - \#)^*$ to Σ^*.

DEFINITION 1. *A* quantum Turing Machine *(QTM) is defined by a triplet* (Σ, Q, δ)*, where* Σ *is a finite alphabet with an identified blank symbol #,* Q *is a finite set of states with an identified initial state* q_0 *and final state* q_f*, and* δ*, the* quantum finite state control *is a function*

$$\delta \ : \ Q \times \Sigma \rightarrow \widetilde{\mathcal{C}}^{\Sigma \times Q \times D}$$

with $D = \{L, R\}$.[2] *The quantum Turing Machine has a two-way infinite tape of cells indexed by the integers* $\mathbf{Z}$*, and by a single read/write tape head that steps along the tape.*

The state of a quantum Turing Machine is a linear superposition $\sum_c \alpha_c |a, q, m\rangle$ over classical configurations $c = |a, q, m\rangle$, where c is a classical configuration consisting of a tape configuration a, a state $q \in Q$, and $m \in \mathbf{Z}$. As in the case of classical Turing Machines, we will consider only those tape configurations a in which all but finitely many tape cells contain the blank symbol #. Also, we will restrict ourselves to considering superpositions consisting of finite linear combinations of configurations c. Formally this restricts us to the dense subspace $\mathcal{S}$ of the Hilbert space with a basis element for each configuration $|c\rangle$. The value $\delta(p, \sigma)$ gives the superposition of updates which the machine will take when in state p reading a σ. In this manner, the transition function δ specifies a linear operator on the space $\mathcal{S}$. Not every transition function δ specifies a legal quantum Turing Machine. For the quantum Turing Machine to be legal, the linear operator specified by δ must be unitary. Such transition functions δ are called well-formed. If the state of the quantum Turing Machine is the superposition $\sum_c \alpha_c |c\rangle$, then performing a measurement on the state of the quantum Turing Machine (in the computational basis) yields the result c with probability $|\alpha_c|^2$.

An Efficient Universal QTM

Unlike in the classical case, the construction of an efficient universal QTM is quite non-trivial. Here we will sketch the main steps in the construction of [**9**].

In order to show that there is an efficient universal quantum Turing Machine, we must show that there is a fixed quantum Turing Machine with a fixed alphabet Σ and fixed set of states Q, that can simulate an arbitrary quantum Turing

[2]By $\widetilde{\mathcal{C}}$, we mean all those complex numbers $x + iy$ where the j-th bit of x and y can be computed in time polynomial in j.

Machine, with possibly a much larger tape alphabet and set of states. To do this, we must allow the universal quantum Turing Machine to encode each symbol of M by a block of symbols. Thus we will assume that there is an efficient encoding and decoding scheme to translate between configurations M and those of the universal quantum Turing Machine. We must also allow the universal quantum Turing Machine several computational steps to simulate each step of M. Finally, we will allow the simulation to be ϵ faithful, i.e. for every input x, and every t, the probability distribution that results from measuring the state of M on input x after t steps must be with in ϵ total variation distance (L_1 norm) of the corresponding probability distribution defined by the simulation on the universal QTM.

DEFINITION 2. *We say that a quantum Turing Machine is in* standard form *if all transitions leading to state $q \in Q$ cause the read/write head to move in direction d.*

For quantum Turing Machines in the standard form, the transition function can be simplified to

$$\delta \ : \ Q \times \Sigma \rightarrow \widetilde{\mathcal{C}}^{\Sigma \times Q}$$

This is because the direction d is uniquely determined by the new state $q \in Q$. It is not hard to see that a quantum Turing Machine in standard form is well-formed if and only if its transition function δ is unitary.

In this case, the main problem in constructing the universal quantum Turing Machine lies in implementing a close approximation to a specified unitary transformation δ using only the fixed size of unitary transformation allowed to the universal quantum Turing Machine. Suppose that δ is a d dimensional unitary transformation. The universal quantum Turing Machine accomplishes this by implementing a sequence of very elementary operations:

DEFINITION 3. *An $m \times m$ unitary matrix U is* near-trivial *if it satisfies one of the following two conditions:*

- *U is the identity except that one of its diagonal entries is $e^{i\theta}$ for some $\theta \in [0, 2\pi]$.*
- *U is the identity except that the submatrix in one pair of distinct dimensions i and j is the rotation by some angle $\theta \in [0, 2\pi]$.*

The following theorem provides one of the key ingredients in showing how to implement any arbitrary m dimensional unitary transformation δ on a universal quantum Turing Machine:

THEOREM 1. *Any m dimensional unitary transformation U can be ϵ-approximated by a product of near-trivial matrices. Moreover, there is a deterministic algorithm that outputs such a decomposition in time polynomial in m and $\log 1/\epsilon$.*

Now the simulation of a given quantum Turing Machine M proceeds as follows. Each tape cell (symbol) of M is encoded by a block of cells of the universal quantum Turing Machine. Also, the state $q \in Q$ of M is written out explicitly on the tape of the universal quantum Turing Machine. Now, to simulate a step of M, the universal quantum Turing Machine carries out the unitary transformation δ on the block of cells representing the currently scanned symbol of M, and the block of cells representing the state $q \in Q$ of M. It then updates the scanned symbol by simulating the movement of the tape head of M.

Dealing with non-standard form quantum Turing Machines

The states $q \in Q$ of a quantum Turing Machine $M = (\Sigma, Q, \delta)$ may be regarded as an orthonormal basis for the Hilbert space $\mathcal{C}^{|Q|}$. Consider a new quantum Turing Machine M', whose states B correspond to a different orthonormal basis for this Hilbert space. Then each state $b \in B$ is a linear combination of the states in Q. Finally, if we define the transition function δ' of M' as the natural (linear) extension of δ, then it is easy to see that M' has the same time evolution as M. The key property that facilitates the simulation of quantum Turing Machines that are not in standard form is:

THEOREM 2. *For any quantum Turing Machine $M = (\Sigma, Q, \delta)$, there is a unitary change of basis for $\mathcal{C}^{|Q|}$, such that the resulting equivalent quantum Turing Machine M' is in standard form.*

The only remaining problem in carrying out the construction of a universal quantum Turing Machine is that the start state and final state of M under the basis change in the theorem above, might correspond to superpositions of states in M'. To fix this problem, we can simulate each step of M using three steps — change basis from Q to B, simulate a step of M', change basis back from B to Q.

Multi-tape quantum Turing Machines

The construction of the universal quantum Turing Machine easily extends to multi-tape quantum Turing Machines in standard form. However, the construction given above does not work for multi-tape quantum Turing Machines. The difficulty is that, for multi-tape QTMs, there is no change of basis, in general that yields an equivalent QTM in standard form. Nevertheless, Yao's construction [**31**], showing the polynomial equivalence of quantum Turing Machines and quantum circuits, implies an efficient universal multi-tape QTM. There is a penalty in the simulation overhead, however, that grows as exponentially in the number of tapes of the QTM. (Since the number of tapes is a constant, this still gives a polynomial overhead.) It is an open question whether this simulation overhead is inherent.

2.2. Quantum Circuits. Quantum circuits [**11**] are another abstract model for quantum computers. Yao [**31**] showed that quantum Turing Machines and uniform families of quantum circuits are equivalent in terms of their computing power.

A quantum circuit on m qubits implements a unitary transformation U on the Hilbert space $(\mathcal{C}^2)^{\otimes m}$. A unitary transformation on n qubits is $c-local$ if it operates nontrivially on at most c of the qubits, and preserves the remaining qubits. Quantum computation may be thought of as the study of those unitary transformations that can be realized as a sequence of $c-local$ unitary transformations, i.e., of the form

$$\mathcal{U} = \mathcal{U}_1 \mathcal{U}_2 \cdots \mathcal{U}_k,$$

where the $\mathcal{U}_i$ are $c-local$ and k is bounded by a polynomial in n.

Indeed, it has been shown that we can simplify the picture further, and restrict attention to certain special $2-local$ unitary transformations, also called elementary quantum gates [**7**]:

Rotation

A rotation or phase shift through an angle θ can be represented by the matrix

$$\mathcal{U} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

This can be thought of as rotation of the axes (see Figure 1).

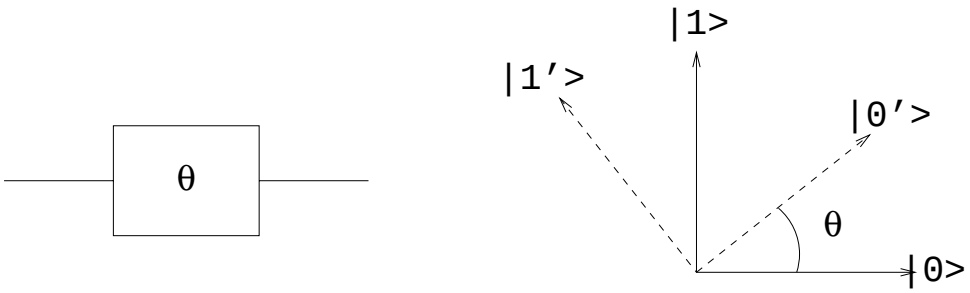

Figure 1. Rotation.

Hadamard transform

If we reflect the axes in the line $\theta = \pi/8$, we get the *Hadamard transform*. (See Figure 2). This can be represented by the matrix

$$\mathcal{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

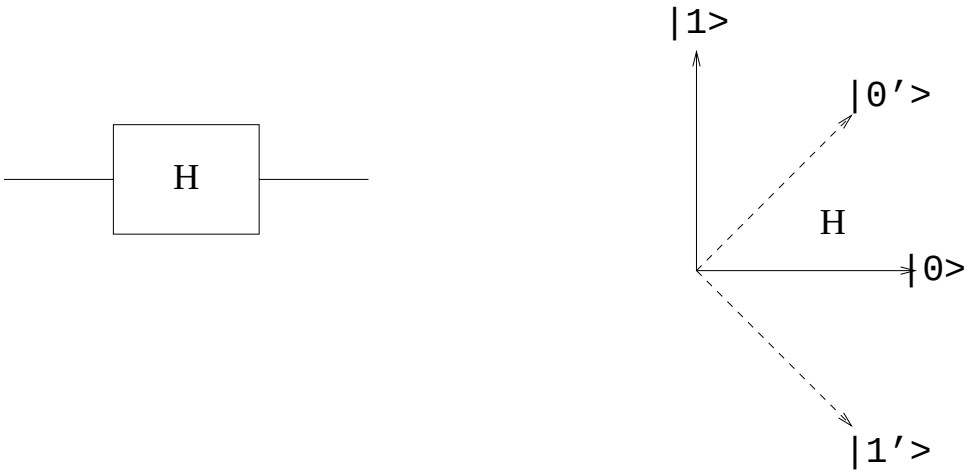

Figure 2. Hadamard transform.

Phase Flip

The phase flip gate operates on a single qubit by the following matrix

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Controlled NOT

The controlled NOT gate (Figure 3) operates on 2 qubits and can be represented by the matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

where the basis elements are (in order) $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$. If inputs a and b are basis states, then the outputs are a and $a \oplus b$.

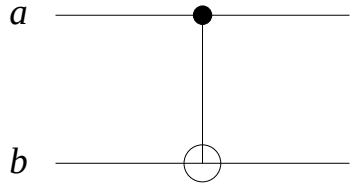

Figure 3. Controlled NOT.

Any unitary transformation can be approximated by using rotation by $\pi/8$, the Hadamard transform, phase flip and controlled NOT gates [**7**].

Primitives for quantum computation

Let us consider how we can simulate a classical circuit with a quantum circuit. The first observation is that quantum evolution is unitary, and therefore reversible. (The effect of a unitary transformation U can be undone by applying its adjoint). Therefore, if the classical circuit computes the function $f : \{0,1\}^n \to \{0,1\}$, then we must allow the quantum circuit to output $|x\rangle |f(x)\rangle$ on input $|x\rangle$. It follows from the work of Bennett [**4**] on reversible classical computation that, if $C(f)$ is the size of the smallest classical circuit that computes f, then there exists a quantum circuit of size $O(C(f))$ which, for each input x to f, computes the following unitary transformation U_f on m qubits

$$U_f : |x\rangle |y\rangle \to |x\rangle |y \oplus f(x)\rangle$$

In general though, we don't need to feed U_f a classical state $|x\rangle$. If we feed U_f a superposition

$$\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle |0\rangle$$

then, by linearity,

$$U_f \left(\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle |0\rangle \right) = \sum_{x \in \{0,1\}^n} \alpha_x U_f (|x\rangle |0\rangle) = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle |f(x)\rangle .$$

At first sight it might seem that we have computed $f(x)$ simultaneously for each basis state $|x\rangle$ in the superposition. However, were we to make a measurement, we would observe $f(x)$ for only one value of x.

Hadamard Transform

The second primitive is the Hadamard transform H_{2^n}, which corresponds to the Fourier transform over the abelian group Z_2^n. The correspondence is based on identifying $\{0,1\}^n$ with Z_2^n, where the group operation is bitwise addition modulo 2. One way to define H_{2^n} is as the $2^n \times 2^n$ matrix in which the (x,y) entry is $2^{-n/2}(-1)^{x\cdot y}$. An equivalent way is as follows. Let H be the unitary transformation on one qubit defined by the matrix

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}.$$

If a quantum circuit consists of subcircuits operating on disjoint qubits, then the unitary transformation describing the circuit is the tensor product of the unitary transformations describing the individual subcircuits. Thus, $H_{2^n} = H^{\otimes n}$, or H tensored with itself n times.

Applying the Hadamard transform (or the Fourier transform over Z_2^n) to the state of all zeros gives an equal superposition over all 2^n states

$$\mathcal{H}_{2^n}|0\cdots 0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle .$$

In general, applying the Hadamard transform to the computational basis state $|u\rangle$ yields:

$$\mathcal{H}_{2^n}|u\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{u\cdot x}|x\rangle$$

2.3. Polynomial Equivalence of Quantum Turing Machines and Quantum Circuits. The central issues to be addressed in constructing an efficient quantum circuit equivalent to a given QTM are the same as those in constructing a universal QTM. In each case, we must decompose one step of the simulated machine (which is a mapping from the computational basis to a new orthonormal basis) into many simple steps, each of which can only map some of the computational basis vectors to their desired destinations. In general, this partial transformation will not be unitary, because the destination vectors will not be orthogonal to the computational basis vectors which have not yet been operated on. The main idea in Yao's construction is to create a second copy of the space, and in each step to map some of the computational basis vectors to their desired destinations, but in the second copy. Once all of the basis vectors have been mapped, then the copies of the space can be interchanged.

THEOREM 3. *A k tape QTM running for T steps can be simulated by a quantum circuit with accuracy ϵ, and size $O(T^2 \log^{O(1)} \epsilon)$.*

PROOF. Let $M = (\Sigma, Q, \delta)$ be a well-formed generalized QTM, and fix $T, \epsilon > 0$.

The simulation uses an extra qubit b, to create two copies of the space of configurations of M. Now if U denotes the unitary transformation carried out by M, then the goal of the simulation is to carry out the transformation $|1\rangle\langle 0|U + |0\rangle\langle 1|U^{-1}$.

The main principle that enables the simulation to be carried out is the following. If U is a unitary transformation acting on the space V, and if $\{P_i\}$ are mutually orthogonal subspaces of V, then the transformation $|1\rangle\langle 0|U + |0\rangle\langle 1|U^{-1}$ on $\mathcal{C}^{\in} \otimes \mathcal{V}$

maps $|0\rangle \otimes P_i$ to $|1\rangle \otimes U(P_i)$ and vice versa. For the simulation of M, the subspaces P_i is spanned by all those configurations of M in which the tape head is in tape cell i. Since the action of the QTM is local, we may write the transformation $|1\rangle\langle 0| U + |0\rangle\langle 1| U^{-1}$ acting on P_i, as a unitary transformation X_i on the smaller set of qubits describing the contents of the three tape cells $i-1, i, i+1$, the finite state control Σ, and the extra qubit b. Since the tape head must be confined between tape cells $-(T-1)$ and $T-1$ in the first T steps, to simulate a step of M, it suffices to carry out the sequence of unitary transformations X_i for i ranging from $-(T-1)$ to $T-1$. Finally, the qubit b must be flipped to restore the state to the original half of the space.

To finish the proof, we observe that each of the transformations X_i is defined on a bounded number of qubits. To simulate them to with in a constant ϵ, we must invoke a theorem of Kitaev and Solovay (see [**22**] for an exposition), which states that if a set of transformations is dense in $SU(2)$ and closed under Hermitian conjugation, then any single qubit gate can be approximated to within ϵ by a product of $\log^{O(1)} \epsilon$ transformations from the set. □

Solovay and Yao [**26**] point out that the same technique can be used to simulate a multi-tape QTM on a single-tape QTM. Instead of building a quantum gate that operates on three adjacent cells to update any configuration with the tape head in the middle cell, we build a quantum gate that operates on sets of three adjacent cells from each of the tapes to update any configuration with each of the tape heads in the middle of the three cells. Since we have $\Theta(T)$ cells for each tape, this means we need to apply the quantum gate $\Theta(T^k)$ times to handle every possible combination of tape head positions. This proof technique therefore only allows a QTM with a constant number of tapes to be simulated with a polynomial slowdown.

Bernstein [**5**] showed how to improve the simulation overhead from T^k to 3^k. It is an open question whether this exponential dependence on k is necessary.

3. Quantum Complexity Classes:

Recall that a language $L \subseteq \Sigma^*$ is in the class P (polynomial time), if there is a polynomial time deterministic Turing Machine which on input $x \in \Sigma^*$ decides whether or not x is in L. The class of efficiently solvable problems is identified with the class BPP (bounded-error probabilistic polynomial time).

Definition: A language $L \subseteq \Sigma^*$ is in the class BPP if there is a probabilistic polynomial time Turing Machine which on input $x \in \Sigma^*$ accepts with probability at least $2/3$ when $x \in L$, and rejects with probability at least $2/3$ when $x \notin L$. The probability of correct answer $2/3$ can be boosted to $1-\epsilon$ by running the probabilistic Turing Machine $O(\log 1/\epsilon)$ times and taking the majority answer.

The class of efficiently solvable problems on a quantum computer is BQP (bounded-error quantum polynomial time).

Definition: A language $L \subseteq \Sigma^*$ is in the class BQP if there is a polynomial time quantum Turing Machine which on input $x \in \Sigma^*$ accepts with probability at least $2/3$ when $x \in L$, and rejects with probability at least $2/3$ when $x \notin L$. Once again the probability of correct answer, $2/3$, can be boosted to $1-\epsilon$ by running the probabilistic Turing Machine $O(\log 1/\epsilon)$ times and taking the majority answer.

BQP may be equivalently defined as the class of languages accepted by a polynomial time uniform family of polynomial size quantum circuits.

Since well-formed quantum Turing Machines must have unitary time evolution, it is not a priori clear that BQP contains P. To show this, one must appeal to work by Bennett [**4**] showing that the class P can be recognized by reversible polynomial time Turing Machines.

Furthermore, a probabilistic Turing Machine can be simulated by a quantum Turing Machine as follows. If the probabilistic Turing Machine M flips k coins, then the quantum Turing Machine starts with a block of k qubits initially in the state $\left|0^k\right\rangle$. Now it performs a $\pi/4$ rotation on each qubit to end up in the state $(\frac{|0\rangle+|1\rangle}{\sqrt{2}})^k = 2^{-k/2}\sum_{x\in\{0,1\}^k}|x\rangle$. Now, if the quantum Turing Machine simulates M, using these k bits as the outcomes of the coin flips (but otherwise leaving these k bits unchanged), then the state at the end of this computation is $2^{-k/2}\sum_{x\in\{0,1\}^k}|x\rangle\,|M(x)\rangle$. Measuring the resulting state yields $M(x)$ for a uniformly random value of x as desired.

Thus we have the containments $P \subseteq BPP \subseteq BQP$.

Next, we show that $BQP \subseteq P^{\#P}$. Recall that, given a predicate $f(x,y)$ that can be computed in time polynomial in $|x|$, the corresponding counting function $g(x) = |\{y : f(x,y)\}|$ is in $P^{\#P}$.

THEOREM 4. *$P \subseteq BPP \subseteq BQP \subseteq P^{\#P} \subseteq PSPACE$.*

We give a sketch of the proof that $BQP \subseteq P^{\#P}$. It uses the fact from [**9**] that we can assume without loss of generality that all the transition amplitudes specified in the transition function δ are real. The action of the transition function δ of QTM M may be described by a tree. The root of the tree corresponds to the initial configuration, and applying δ to the configuration corresponding to any node yields a superposition of configurations represented by the children of that node. The leaves of the tree correspond to accepting and rejecting configurations. Let p be a root-leaf path in this tree. The amplitude of this path β_p is just the product of the branching amplitudes along the path, and is computable to within $1/2^j$ in time polynomial in j. Several paths may lead to the same configuration c. Thus the amplitude of c after T steps of computation is the following sum over all T length paths p: $\alpha_c = \sum_{p\ to\ c}\beta_p$. The probability that QTM M accepts is $\sum_{accepting\ c}|\alpha_c|^2$. Let $a_p = max(\beta_p, 0)$ and $b_p = max(-\beta_p, 0)$. Then $|\alpha_c|^2$ can be written as $|\alpha_c|^2 = \sum_{p\ to\ c}(a_p - b_p)^2 = \sum_{p\ to\ c} a_p^2 + b_p^2 - \sum_{p,p'\ to\ c} 2a_pb_p$. It follows that the acceptance probability of M can be written as the difference between the two quantities $\sum_{accepting\ c}\sum_{p\ to\ c} a_p^2 + b_p^2$, and $\sum_{accepting\ c}\sum_{p,p'\ to\ c} 2a_pb_{p'}$. Since each of these quantities is easily seen to be in $P^{\#P}$, it follows that $BQP \subseteq P^{\#P}$.

In view of this theorem, we cannot expect to prove that BQP strictly contains BPP without resolving the long standing open question in computational complexity theory, namely, whether or not $P = PSPACE$.

4. Recursive Fourier Sampling

In this section, we shall introduce the recursive Fourier sampling problem, show that it can be solved in polynomial time on a quantum computer. However, we shall

also show that it does not even lie in the class MA (the probabilistic generalization of NP, see section 6 for a definition) relative to some oracle.

The recursive Fourier sampling problem was defined by Bernstein and Vazirani [**9**], and was inspired by earlier work by Deutsch and Jozsa [**13**]. It not only provided the first example of a problem that demonstrates a superpolynomial speedup of quantum algorithms over probabilistic algorithms, but it also shows that $BQP \not\subseteq MA$ relative to an oracle. The proof presented here is a modification of the proof from [**9**], followed by an elegant argument from [**17**].

First we introduce the basic primitive, Fourier sampling.

Fourier Sampling

In general, if we start with a state $|\phi\rangle = \sum_x \alpha_x |x\rangle$, then, after applying the Fourier transform over Z_2^n, we obtain the new state $\left|\widehat{\phi}\right\rangle = \sum_x \alpha_x |\widehat{x}\rangle$. Notice that this transform can be computed by applying only n single qubit gates, whereas it is computing the Fourier transform on a 2^n dimensional vector. However, the output of the Fourier transform is not accessible to us. To read out the answer, we must make a measurement, and now we obtain x with probability $|\widehat{\alpha}_x|^2$. This process of computing the Fourier transform and then performing a measurement is called Fourier sampling, and is one of the basic primitives in quantum computation.

To see the power of Fourier sampling, suppose we are given a function $f : \{0,1\}^n \to \{1,-1\}$ such that there is an $s \in \{0,1\}^n$ such that for all x, $f(x) = s \cdot x$, where $s \cdot x$ denotes the dot product $s_1x_1 + \cdots + s_nx_n \bmod 2$, i.e., f is one of the Fourier basis functions. The task is to determine which one it is. The following quantum algorithm carries out this task using two quantum registers, the first consisting of n qubits, and the second consisting of a single qubit.

- Start with the registers in the state $|0^n\rangle |0\rangle$
- Compute the Fourier transform on the first register to get $\sum_{x\in\{0,1\}^n} |x\rangle \otimes |0\rangle$.
- Compute f to get $\sum_x |x\rangle |f(x)\rangle$.
- Apply a conditional phase based on $f(x)$ to get $\sum_x (-1)^{f(x)} |x\rangle |f(x)\rangle$.
- Uncompute f to get $\sum_x (-1)^{f(x)} |x\rangle \otimes |0\rangle$.
- Compute the Fourier transform on the first register to get $|s\rangle \otimes |0\rangle$.

Measuring the first register now yields the string s. Notice that the quantum algorithm queried the boolean function f only twice, but obtained n bits of information about it in the process! An easy information theoretic argument shows that any classical (probabilistic) algorithm must make at least n queries to obtain this information. This single iteration of Fourier sampling is carried out by exactly the quantum circuit devised by Deutsch and Jozsa [**13**]. The only difference here is that, unlike the Deutsch-Jozsa problem, here all the n output bits are measured (and the promise that f must satisfy is different).

One way to use this difference in the number of queries in order to demonstrate a gap between quantum and probabilistic algorithms is to make the queries very expensive. Then the quantum algorithm would be $n/2$ times faster than any probabilistic algorithm for the given task. The idea behind proving a superpolynomial gap is to make each query itself be the answer to a Fourier sampling problem. Now each query itself is much easier for the quantum algorithm than for any probabilistic

algorithm. Carrying this out recursively for $\log n$ levels leads to the superpolynomial speedup for quantum algorithms.

Recursive Fourier sampling

We say that a boolean function f defines a Fourier sampling tree if $f(x_1, \ldots, x_k) = s(x_1, \ldots, x_{k-1}) \cdot x_k$, where $\cdot$ denotes the dot product as defined above. We say that f is *derived from* g (and g *specifies* the Fourier sampling tree f) if $f(x_1, \ldots, x_{k-1}) = g(x_1, \ldots, x_{k-1}, s)$. In the recursive Fourier sampling problem, the x_j's form a telescoping series, so that $|x_{j+1}| = |x_j|/2 = 2^{l-j}$, down to $|x_{l+1}| = 1$. Now, given an oracle for g, and the values of f for the leaf nodes (of the form $(x_1, x_2, \ldots, x_{l+1})$), the challenge is to determine $f(x_1)$.

A recursive implementation of the Fourier sampling primitive introduced above can be used to give a polynomial time quantum algorithm to solve this problem as follows. For the base case, since f is known at the leaf nodes, to determine $f(x_1, x_2, \ldots, x_l)$, we simply perform Fourier sampling with respect to the two leaves in this subtree, to determine $s(x_1, x_2, \ldots, x_l)$ and then output $g(x_1, x_2, \ldots, x_l, s)$. In general, to determine $f(x_1, \ldots, x_{k-1})$, we perform Fourier sampling (recursively using the values of $f(x_1, \ldots, x_{k-1}, x_k)$) to determine $s(x_1, \ldots, x_{k-1})$, and then output $g(x_1, \ldots, x_{k-1}, s)$. Since the Fourier sampling requires only two queries to $f(x_1, \ldots, x_{k-1}, x_k)$, the running time of this quantum algorithm satisfies the recurrence relation $T(n) \leq 2T(\frac{n}{2}) + O(n)$, whose solution is $T(n) = O(n \log n)$.

On the other hand, if a classical algorithm were to take the same kind of recursive approach, it must make n queries at the highest level, each requiring $n/2$ queries at the next level, etc. The recurrence relation is now the following — $T(n) \leq nT(\frac{n}{2}) + \Omega(n)$, which implies $T(n) = \Omega(n^{\log n})$. Proving such a lower bound for any probabilistic algorithm (and generalizing it to any Merlin-Arthur protocol) requires more work. We start by proving that if g is chosen uniformly at random from among functions specifying Fourier sampling trees f, and we condition on the values of g at any choice of $o(n^{\log n})$ points, then the value of $f(x_1)$ is almost unbiased.

Consider a set S of query-answer pairs that are consistent with some legal Fourier sampling tree. Consider a node $y = (x_1, \ldots, x_{k-1})$ in the tree. Consider a legal Fourier sampling tree f which agrees with S below y in the tree. Then f determines string s such that $f(x_1, \ldots, x_k) = x_k \cdot s$. Say that y is a hit if $(x_1, \ldots, x_{k-1}, s)$ is a query in S. Let $P(y)$ denote the probability that y is a hit (with respect to S) when g is chosen uniformly among all functions such that the derived Fourier sampling tree f agrees with S below y in the tree. Similarly, if z is an ancestor of y, denote by $P_z(y)$ the probability that y is a hit when g is chosen uniformly among all functions such that the derived Fourier sampling tree f agrees with S below z in the tree.

LEMMA 1. *$P_z(y) \leq 2P(y)$.*

Proof Sketch We observe that the queries outside of the subtree rooted at y can only determine the value of $f(y)$, but not the string s. If y is not a hit, then the only constraint this places is on $g(y, s)$. The lemma follows.

LEMMA 2. *If there are q queries of S in the subtree rooted at y, then $P_x(y) \leq q/\gamma(n/4)$, where $n = 2^\ell$, and ℓ is the height of y in the tree. Here $\gamma(n)$ is the product $n(n/2)\cdots 1 = 2^{\ell(\ell+1)/2}$.*

Proof Sketch The proof is by induction on ℓ. For the inductive step, if q' of the q queries are at y, and there are exactly c hits among the 2^n children of y, then the probability that y is a hit is at most $\frac{q'}{2^{n-c}}$. Thus the probability that y is a hit is bounded above by the sum of $\frac{q'}{2^{n/2}}$ and the probability that at least $n/2$ of the children of y are hits. By the induction hypothesis and an easy manipulation the latter probability, it can be bounded by $\frac{q-q'}{2\gamma(n/4)}$. It follows that $P(y) \leq \frac{q-q'}{2\gamma(n/4)} + \frac{q'}{2^{n/2}} \leq \frac{q}{2\gamma(n/4)}$.

By the above lemma, it follows that any deterministic algorithm that makes $o(n^{\log n})$ queries gives the wrong answer on at least $1/2 - o(1)$ fraction of inputs g chosen uniformly at random from among functions specifying Fourier sampling trees f. Now by Yao's lemma [**30**], it follows that any probabilistic algorithm that makes $o(n^{\log n})$ queries must have error probability at least $1/2 - o(1)$ on some input. This completes the proof that relative to some oracle, BQP properly contains BPP.

To show that relative to some oracle BQP contains a problem that is not in MA, we start with the following observation. The above lemma actually shows that any non-deterministic algorithm that makes $o(n^{\log n})$ queries gives the wrong answer on at least $1/2 - o(1)$ fraction of inputs g chosen uniformly at random from among functions specifying Fourier sampling trees f.

Now we show that there is an oracle relative to which the recursive Fourier sampling problem is not in MA. The proof is by contradiction.

Assume to the contrary that there is a Merlin-Arthur protocol that runs in time $poly(n)$. Then if the input g is in the language, there is a proof of length at most $poly(n)$ such that the verifier accepts with probability at least $2/3$ (and running time at most $poly(n)$). And if the input g is not in the language, then for every proof the verifier rejects with probability at least $2/3$. Since the protocol is Merlin-Arthur, we can boost the success probability by running the verifier's algorithm several times using independent random strings (without increasing the length of the proof that prover sends). Thus by increasing the running time of the verifier to $O(poly(n) \times poly'(n))$, we can replace the error probability $1/3$ above by $\frac{1}{2^{poly'(n)}}$. Now let us pick g uniformly at random from among functions specifying Fourier sampling trees f. Then if g is in the language, clearly the verifier rejects with probability at most $\frac{1}{2^{poly'(n)}}$. If g is not in the language, then the verifier accepts each possible proof with probability at most $\frac{1}{2^{poly'(n)}}$. But since there are at most $2^{poly(n)}$ proofs of length $poly(n)$, it follows that the verifier accepts with probability at most $\frac{2^{poly(n)}}{2^{poly'(n)}}$. Thus in each case the verifier errs with probability at most $\frac{2^{poly(n)}}{2^{poly'(n)}}$. It follows that there is a fixed random tape for the verifier such that the verifier errs on at most $\frac{2^{poly(n)}}{2^{poly'(n)}}$ fraction of the g's. But with a fixed random tape, the verifier is deterministic, and this contradicts the consequence of the lemma above that any non-deterministic algorithm that makes $o(n^{\log n})$ queries must give the wrong answer on at least $1/2 - o(1)$ fraction of inputs g. This completes the proof of the following theorem:

THEOREM 5. *There is an oracle relative to which $BQP \not\subseteq MA$.*

5. Quantum Lower Bounds

In view of the exponential advantage offered by quantum algorithms for certain computational problems such as prime factorization, it is tempting to ask whether quantum computers can solve NP-complete problems in polynomial time.

Consider the NP-complete problem — 3-SAT. Given a $3CNF$ Boolean formula $f(x_1, x_2, \ldots, x_n)$, is there an boolean assignment to $x_1, \ldots, x_n$ that satisfies f? Brute force search would take $O(N)$ steps, where $N = 2^n$. Is it possible that the exponential advantage of quantum computers could be used to determine such a truth assignment in polynomial time?

We will abstract this problem in a black box or oracle model as follows. Assume that the input to the problem is a table with N boolean entries, and the task is to find whether any entry in the table is 1. Classical algorithms are allowed random access to the table entries, and quantum algorithms can query the entries of the table in superposition, i.e., a query is of the form $\sum_x \alpha_x |x\rangle |0\rangle$, and the answer to the query is $\sum_x \alpha_x |x\rangle |f(x)\rangle$. It was shown in [**10**] that any quantum algorithm for this problem must make $\Omega(\sqrt{N})$ queries to the table. Their proof is based on a technique called the hybrid argument. In view of this result, resolving the question whether $NP \subseteq BQP$ will require a non-relativizing proof technique, and is not likely barring a major advance in computational complexity theory. There is a striking algorithm due to Grover [**15**] that provides a matching upper bound.

A second technique for proving quantum lower bounds — the method of polynomials — was introduced in [**8**]. It is based on the fact that the acceptance probability of a quantum algorithm after T queries to the input can be described by a polynomial of degree $2T$. The lower bound on a specific function f is proved by showing that it cannot be approximated by a polynomial of degree $2T$ unless T is very large. This method was used, for example, to give a tight linear lower bound for the quantum complexity of the parity function. They also proved that for any total function, the deterministic query complexity $D(h) = O(Q(h)^6)$, where $Q(h)$ is the quantum query complexity. In this section, we sketch a simpler proof of this result using the hybrid argument.

Recently, a third technique for proving lower bounds, the method of quantum adversaries [**2**] has been introduced. We illustrate the quantum adversary technique by re-proving the $\Omega(2^{n/2})$ lower bound for the search problem using this technique. This technique appears very general, and indeed has been used to obtain tight bounds on the query complexity of a number of problems.

A notable example is the following problem. Let $f : \{0,1\}^n \to \{0,1\}^n$ be a random permutation. Find $f^{-1}(0^n)$. The best previous bound on this problem was $\Omega(2^{n/3})$, via the hybrid argument [**10**]. However, [**2**] gives a tight $\Omega(2^{n/2})$ lower bound using this quantum adversary technique.

There are still important open questions in this area of quantum lower bounds. Perhaps the most striking is the following: Let $f : \{0,1\}^n \to \{0,1\}^n$ be a $2-1$ function. Find x and y such that $f(x) = f(y)$. No non-trivial lower bound is known for this problem. An efficient algorithm for this problem would imply that quantum computation precludes the possibility of collision-intractible hashing — an important cryptographic primitive.

5.1. The hybrid argument.

THEOREM 6. *Any quantum algorithm, in the black box model, for determining whether there exist $x_1, \ldots, x_n$ such that $f(x_1, \ldots, x_n) = 1$ must make $\Omega(\sqrt{N})$ queries to f.*

PROOF. Consider any quantum algorithm A for solving this search problem. First do a test run of A on function $f \equiv 0$. Define the query magnitude of x to be $\sum_t |\alpha_{x,t}|^2$, where $\alpha_{x,t}$ is the amplitude with which A queries x at time t. The expectation value of the query magnitudes $E_x\left(\sum_t |\alpha_{x,t}|^2\right) = T/N$. Thus $min_x\left(\sum_t |\alpha_{x,t}|^2\right) \leq T/N$. If the minimum occurs at z, then by the Cauchy-Schwarz inequality $\sum_t |\alpha_{z,t}| \leq T/\sqrt{N}$.

Let $|\phi_t\rangle$ be the states of A_f after the t-th step. Now run the algorithm A on the function $g : g(z) = 1, g(y) = 0 \forall y \neq z$. Suppose the final state of A_g is $|\psi_T\rangle$. By the claim that follows, $|| \, |\phi_T\rangle - |\psi_T\rangle \, || \leq \sum_t |\alpha_{z,t}| \leq T/\sqrt{N}$. This implies that the two states can be distinguished with probability at most $O(T/\sqrt{N})$ by any measurement. Thus any quantum algorithm that distinguishes f from g with constant probability of success must make $T = \Omega(\sqrt{N})$ queries. □

CLAIM 1. *$|\psi_T\rangle = |\phi_T\rangle + |E_0\rangle + |E_1\rangle + \ldots + |E_{T-1}\rangle$, where $|| \, |E_t\rangle \, || \leq |\alpha_{z,t}|$.*

PROOF. Consider two runs of the algorithm A, which differ only on the t-th step. The first run queries the function f on the first t steps and queries g for the remaining $T - t$ steps; the second run queries f on the first $t - 1$ steps and g for the remaining $T - t + 1$ steps. Then at the end of the t-th step, the state of the first run is $|\phi_t\rangle$, whereas the state of the second run is $|\phi_t\rangle + |F_t\rangle$, where $|| \, |F_t\rangle \, || \leq |\alpha_{x,t}|$. Now if U is the unitary transform describing the remaining $T - t$ steps (of both runs), then the final state after T steps for the two runs are $U \, |\phi_t\rangle$ and $U(|\phi_t\rangle + |F_t\rangle)$, respectively. The latter state can be written as $U \, |\phi_t\rangle + |E_t\rangle$, where $|E_t\rangle = U \, |F_t\rangle$. Thus the effect of switching the queried function only on the t-th step can be described by an "error" $|E_t\rangle$ in the final state of the algorithm, where $\||E_t\rangle\| \leq |\alpha_{z,t}|$.

We can transform the run A_f to A_g by a succession of T changes of the kind described above. Therefore, by the linearity of quantum mechanics, the difference between the final states of A_f and A_g is $|E_0\rangle + |E_1\rangle + \ldots + |E_{T-1}\rangle$, where $\||E_t\rangle\| \leq |\alpha_{z,t}|$. □

5.2. Block Sensitivity and the Black Box Model.

In this section we will work more explicitly in the black box model, where the algorithm has to explicitly query the input $w \in \{0,1\}^n$, and the complexity of the algorithm is the number of queries that the algorithm makes. We give a simple proof via the hybrid argument of the result of [**8**] showing that $D(f) = O(Q(f)^6)$, where $D(f)$ is the deterministic query complexity of f and $Q(f)$ is its quantum query complexity:

For a boolean function $f : \{0,1\}^n \to \{0,1\}$, $D(f)$ is the minimum number of bits of the input that a deterministic algorithm must query to compute f. $Q(f)$ is the minimum number of queries to the input w that a quantum algorithm must make to compute $f(w)$ with error probability at most $1/3$.

$C(f)$, the certificate complexity or the nondeterministic complexity of f, is the minimum number of bits of the input that must be revealed (by someone who knows all the input bits) to convince a deterministic algorithm about the value of $f(w)$.

A key result, that was first discovered by Blum [**6**], shows that in the black box model the deterministic and nondeterministic (certificate) complexity of a function are polynomially related. Recall that in the black box model we only count the number of queries made by the algorithm, not the number of steps of computation performed by the algorithm between queries. In fact, the $C(f)^2$ upper bound on the deterministic complexity is established by giving an algorithm that requires $2^{C(f)}$ steps of computation, but only $C(f)^2$ queries.

LEMMA 3. *$C(f) \leq D(f) \leq C(f)^2$.*

Another closely related property is a structural property of f called its block sensitivity:

Notation: For a string $w \in \{0,1\}^n$ and a set $S \subseteq \{1,2,\ldots,n\}$, define $w^{(S)}$ to be the boolean string y that differs from w on exactly the bit positions in the set S.

DEFINITION 4. *For a boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ the block sensitivity of f, $bs(f)$ is defined to be the maximum number t such that there exists an input $w \in \{0,1\}^n$ and t disjoint subsets $S_1, \ldots S_t \subseteq \{1,2,\ldots,n\}$ such that for all $1 \leq i \leq t$, $f(w) \neq f(w^{(S_i)})$.*

Nisan [**21**] proved the following fundamental lemma:

LEMMA 4. *$\sqrt{C(f)} \leq bs(f) \leq C(f)$.*

COROLLARY 1. *$D(f) \leq bs(f)^4$.*

[**8**] improve this bound by showing:

LEMMA 5. *$D(f) \leq C(f)bs(f) \leq bs(f)^3$.*

We are now ready to prove the relationship between $D(f)$ and $Q(f)$:

THEOREM 7. *$Q(f) = \Omega(\sqrt{bs(f)})$.*

PROOF. The proof mirrors the $\sqrt{N}$ lower bound from the last section. Let $w \in \{0,1\}^n$ be the input defining the block sensitivity of f. Given a quantum algorithm A define the query magnitude of set $S \subseteq \{1,2,\ldots,n\}$ on input w to be $q_S(w) = sum_{j \in S} q_j(w)$. Now if A runs for T steps, then since the sets S_i are disjoint, the expected query magnitude for a random set S_i is $q_{S_i}(w) \leq T/bs(f)$. Let $z = w^{(S_i)}$, where $q_{S_i}(w)$ is the minimum among these query magnitudes. Let ϕ_T and ψ_T denote the final states on inputs w and z, respectively. Then, as in the $\sqrt{N}$ lower bound from the previous subsection, $|| \, |\psi_T\rangle - |\phi_T\rangle \, || \leq T/\sqrt{bs(f)}$. Since A must distinguish these states with constant probability, it follows that $T = \Omega(\sqrt{bs(f)})$. □

COROLLARY 2. *$D(f) = O(Q(f)^6)$.*

5.3. The Method of Quantum Adversaries. In this section we illustrate the method of quantum adversaries, by proving the lowerbound for unstructured search: given a boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$, is there an $x \in \{0,1\}^n$ such that $f(x) = 1$?

We assume that the quantum computer is partitioned into two registers, the input register (specifying the function f) and the work space register. Now, for the search problem, the input function f may be identified for convenience by x, the unique n bit string on which $f(x) = 1$. Thus, the initial state of the registers may be written as $|x\rangle \, \|0\rangle$. If the quantum algorithm correctly solves the search problem, when it is done the work space must look like $|x\rangle \, |x\rangle \, |junk_x\rangle$. But now if the

algorithm works correctly on every input, it must work correctly on a superposition of inputs as well. In the case of the search problem, if the input is a uniform superposition over $|x\rangle$, then the initial state is $(\sum_x |x\rangle) \otimes |0\rangle$, and the final state is $\sum_x |x\rangle \otimes (|x\rangle\,|junk_x\rangle)$. The main point of the quantum adversary argument is that the two registers are initially unentangled, whereas they are maximally entangled at the end of the algorithm. Now establishing an upper bound on the increase in entanglement per step of the quantum algorithm implies a corresponding lower bound on the number of steps that the quantum algorithm must take.

The first step towards fleshing out the above argument is to quantify how entangled the two registers are. Consider the density matrix that describes the state of the input register when the work register is measured. For the search problem, this density matrix is initially (at the beginning of the algorithm)

$$\rho_0 = \left(\sum_{i=1}^{N} |i\rangle\right)\left(\sum_{i=1}^{N} \langle i|\right) = \begin{pmatrix} \frac{1}{N} & \cdots & \frac{1}{N} \\ \vdots & & \vdots \\ \frac{1}{N} & \cdots & \frac{1}{N} \end{pmatrix}$$

corresponding to the state $1/\sqrt{N}\sum_i |i\rangle$. The final density matrix at the end of the algorithm is

$$\rho_F = \begin{pmatrix} \frac{1}{N} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{N} \end{pmatrix}$$

corresponding to the uniform distribution over all $|i\rangle$.

As a measure of entanglement, consider the sum of absolute values of the off-diagonal entries. Initially, this quantity is 0 and finally it is $(N^2 - N) * \frac{1}{N} = \theta(N)$.

How fast do the off-diagonal entries of the density matrix increase? Each step of a quantum algorithm may be regarded as performing a query on the input tape, followed by a unitary transformation on the work register. Clearly the unitary transformation on the work register does not affect the reduced density matrix describing the input register.

To understand the effect of a query, recall how a query is carried out. The input to be queried is written out on a part of the work register designated to be the query register. (Thus the contents of the register might be $\sum_j \alpha_j |j\rangle$.) The reduced density matrix of the input register at this step (just before the query is actually performed) may be written in the form $\rho = \sum_j p_j \rho_j$, where $p_j = |\alpha_j|^2$.

Each ρ_i can be expanded as $\sum_{j,k} \rho^{(i)}_{j,k} = \sum_j \rho^{(i)}_{i,j} + \sum_k \rho^{(i)}_{k,i} = 2\sum_j \rho^{(i)}_{i,j}$, since all the other entries in the sum cannot occur. If the query register contains $\sum \alpha_i |i\rangle$, the query modifies $\rho_{i,j}$ with probability $\alpha_i \alpha_j$.

To bound the effect of the query on ρ, it suffices to bound its effect on each ρ_j. But since this part of the query accesses the input only on location j, the only entries of ρ_j that change are those in the j-th row and in the j-th column. There are only $O(N)$ such entries, and the maximum is achieved when they are equal, and is therefore bounded by $N * O(\frac{1}{\sqrt{N}}) = O(\sqrt{N})$. Since ρ is a convex combination of the ρ_j, it follows that each query can change the off-diagonal entries of ρ by $O(\sqrt{N})$. Since the total change in the off-diagonal entries is $\Theta(N)$, it follows that any quantum algorithm must take $\Omega(\sqrt{N})$ steps to solve the search problem.

Thus the changes to the density matrix are only to those summands, and they change at most $|\alpha_i| \sum |\alpha_j| = O\left(\sqrt{N}\right)$. Since the starting and ending density matrices differ by $O(\sqrt{N})$, the number of queries required is $O(\sqrt{N})$.

So far we assumed that the quantum algorithm gives the correct answer with certainty. To prove a meaningful lower bound, we must allow the quantum algorithm to give an incorrect answer with a small constant probability ϵ. But it is easy to see that the above argument still works, since it can be easily verified that the total change in the off-diagonal entries of the initial density matrix and the final density matrix is still $\Theta(N)$, since the final density matrix satisfies $|\rho_{ij}| \leq \frac{2\sqrt{\epsilon}(1-\epsilon)}{N}$.

6. Quantum NP

The class NP (non-deterministic polynomial time) contains many thousand of the most important computational problems. Of these problems, the vast majority are NP-complete. This means that these are the hardest problems in NP. By this we mean that, if anyone of them can be solved by a polynomial time algorithm, then every problem in NP can be solved by a polynomial time algorithm. The cornerstone of this theory of NP-completeness is the Cook-Levin theorem, which states that 3-SAT is NP-complete.

A language L is in NP if there is a polynomial time proof checker C and a polynomial $poly$, with the following property: if $x \in L$ then there is a string y with $|y| \leq poly|x|$, such that $C(x, y) = 1$. If $x \notin L$, then for every y such that $|y| \leq poly(|x|)$, $C(x, y) = 0$.

Recently Kitaev [**18**] gave the quantum analogue of the Cook-Levin theorem by showing that QSAT the quantum analogue of 3-SAT is complete for BQNP or QMA. Our exposition of this result is based upon the manuscript by Aharonov and Nave [**1**].

BQNP or QMA is the quantum generalization of MA — the probabilistic analogue of NP. To define MA, we simply replace the deterministic polynomial time proof checker with a probabilistic polynomial time proof checker C. Now if $x \in L$, then there is a string y with $|y| \leq poly|x|$, such that $C(x, y) = 1$ with probability at least $2/3$. If $x \notin L$, then for every y such that $|y| \leq poly(|x|)$, $C(x, y) = 0$ with probability at least $2/3$.

To define BQNP, the quantum analogue of MA, we replace the probabilistic polynomial time proof checker by a quantum polynomial time proof checker. Equally important, the witness string y is now allowed to be a quantum witness, i.e., it can be a superposition over strings of length at most $poly(|x|)$.

BQP is trivially contained in BQNP since it can be simulated by the verifier alone. MA is also contained in BQNP since quantum machines can perform the classical computations of their classical counterparts. Kitaev's proof that QSAT is BQNP-complete implies a non-trivial upper bound, showing that $BQNP \subseteq P^{\#P}$.

A BQNP-Complete Problem

Recall that a Hamiltonian acting on n qubits is a 2^n dimensional Hermitian matrix. Say that a Hamiltonian is c-local if it acts as the identity on all except c of the qubits. Consider the following problem:

DEFINITION 5. ***Local Hamiltonians or Q5SAT:*** *Let H_j (for $j = 1, \ldots r$) be 5-local Hamiltonians on n qubits (each specified by complex $2^5 \times 2^5$ matrices.). Assume that each H_j is scaled so that all eigenvalues λ of H_j satisfy $0 \leq \lambda \leq 1$. Let $H = \sum_{j=1}^{r} H_j$. There is a promise about H that either all eigenvalues of H are $\geq b$ or there is an eigenvalue of H that is $\leq a$, where $0 \leq a < b \leq 1$ and the difference $b - a$ is at least inverse polynomial in n, i.e., $b - a \geq \frac{1}{poly(n)}$. The problem asks whether H has an eigenvalue $\leq a$.*

The Connection with 3-SAT

In 3-SAT, we are given a formula f on n variables in 3-CNF (conjunctive normal form.) That is, f is a conjunction of many clauses c_i:

$$f(x_1, x_2, \ldots, x_n) = c_1 \wedge c_2 \wedge \ldots \wedge c_m \ ,$$

where each clause c_j is a disjunction of three variables or their negations. For example, c_j may be $(x_a \vee \overline{x_b} \vee x_c)$.

We would like to make a corresponding Hamiltonian H_i for each clause c_i. H_i should penalize an assignment which does not satisfy the clause c_i. In the example where $c_j = (x_a \vee \overline{x_b} \vee x_c)$, we want to penalize the assignment state $|010\rangle$. If our notion of *penalize* is to have a positive eigenvalue, then we can let $H_j = |010\rangle \langle 010|$, and define the other H_i's similarly, i.e., each H_i has a 1 eigenvalue with a corresponding eigenvector that causes clause c_i to be false.

Finally, we let

$$H = \sum_{i=1}^{m} H_i,$$

so that H is a sum of 3-local Hamiltonians. It is not hard to see that the smallest eigenvalue of H is the minimum (over all assignments) number of unsatisfied clauses. In particular, H has a 0 eigenvalue exactly when there is a satisfying assignment for f.

For general QSAT instances, the Hamiltonians H_j cannot be simultaneously diagonalized in general, and the problem appears much harder.

Membership in BQNP

We can assume without loss of generality that each H_j is just a projection matrix $|\phi_j\rangle \langle \phi_j| \otimes I$. The prover would like to provide convincing and easily verifiable evidence that $H = \sum H_j = \sum (A_j \otimes I)$ has a small eigenvalue $\lambda \leq a$. The proof consists of (a tensor product of) polynomial in n copies of the corresponding eigenvector η.

$\lambda = \sum_j \langle \eta | H_j | \eta \rangle$. Given a single copy of $|\eta\rangle$, the verifier can flip a coin with bias $\frac{\lambda}{r}$ as follows:

(1) Pick $H_j = |\phi_j\rangle \langle \phi_j|$ at random
(2) Measure $|\eta\rangle$ by projecting onto $|\phi_j\rangle$.

This succeeds with probability $\frac{\lambda}{r}$. Given the promise that $\lambda \leq a$ or $\lambda \geq b$, it suffices for the verifier to repeat this test $\frac{r^2}{(b-a)^2}$ times to conclude with high confidence that $\lambda \leq a$. Thus polynomial in n copies of $|\eta\rangle$ are sufficient. Note that

since the verifier is performing each test randomly and independently, the prover gains no advantage by sending an entangled state to the verifier.

BQNP-Completeness

To show that QSAT is complete in BQNP, we need to show that the universal BQNP problem reduces to it. That is, given a quantum circuit $U = U_L U_{L-1} \dots U_1$ and a promise that exactly one of the following holds:

(1) $\exists |\eta\rangle$, U accepts on input $|\eta\rangle$ with probability $\geq p_1 = 1 - \epsilon$
(2) $\forall |\eta\rangle$, U accepts on input $|\eta\rangle$ with probability $\leq p_0 = \epsilon$,

The challenge is to design an instance of QSAT which allows us to distinguish the above two cases. i.e. we wish to specify a sum of local Hamiltonians that has an eigenvector with small eigenvalue if and only if $\exists |\eta\rangle$ that causes U to accept with high ($\geq p_1$) probability.

The construction of the local Hamiltonian is analogous to Cook's theorem. The quantum analogue of the accepting tableau in Cook's theorem will be the computational history of the quantum circuit:

$$|T\rangle = \sum_{t=0}^{L} |\phi_t\rangle \otimes |t\rangle$$

where $|\phi_0\rangle$ is a valid initial state and $|\phi_i\rangle = U_i |\phi_{i-1}\rangle$. Thus the computation history $|T\rangle$ is an element of $(\mathcal{C}^2)^{\otimes n} \otimes \mathcal{C}^{L+1}$. It is a superposition over time steps of the state of the quantum bits as the quantum circuit operates on them.

Now the idea of the BQNP-completeness proof is to design the Hamiltonian H such that:

(1) if there exists $|\eta\rangle$ where $U |\eta\rangle$ accepts with probability at least $1 - \epsilon$, then the computational history $|T\rangle$ of the quantum circuit U on input η is an eigenvector with eigenvalue at most $\frac{\epsilon}{L+1}$
(2) if U rejects every input with probability at least $1 - \epsilon$, then all the eigenvalues of H are at least $\frac{c(1-\epsilon)}{L^3}$

H will be the sum $H_{initial} + H_{final} + H_{propagate}$. The first two terms are simple and express the condition that the computational history starts with a valid input state, and ends in an accepting state.

We consider the first m bits of U's state the input bits and the remaining $n-m$ bits to be the clean work bits. The design of the $H_{initial}$ component should then reflect that at time 0, all of the work bits are clear:

$$H_{initial} = \sum_{s=m+1}^{n} \Pi_s^{(1)} \otimes |0\rangle \langle 0|$$

where $\Pi_s^{(1)}$ denotes projection onto the s-th qubit with value 1.

Assume that the state of the first qubit at the output determines whether or not the input is accepted. Then H_{final} needs to indicate that at time L the first qubit is a 1:

$$H_{final} = \Pi_1^{(0)} \otimes |L\rangle \langle L| .$$

The most complicated component of H is $H_{propagate}$, which captures transitions between time steps. $H_{propagate} = \sum_{j=1}^{L} H_j$, where

$$\begin{aligned} H_j = &-\frac{1}{2} U_j \otimes |j\rangle \langle j-1| \\ &- \frac{1}{2} U_j^\dagger \otimes |j-1\rangle \langle j| \\ &+ \frac{1}{2} I \otimes (|j\rangle \langle j| + |j-1\rangle \langle j-1|) \end{aligned}$$

The fact that the computational history is a superposition over time steps is quite crucial here. To check that the correct operation has been applied in step j, it suffices to restrict attention to the $j-1$-st and j-th bit of the clock (assuming that the clock is represented in unary). Now the quantum register is in a superposition over its state at time $j-1$ and at time j. Locally checking this superposition is sufficient to determine whether its clock j component is the result of applying the quantum gate U_j to the clock $j-1$ component. This is precisely what the Hamiltonian H_j above is designed to do.

Next we show that an accepting history of computation is an eigenvector of H with eigenvalue 0.

Let $|T\rangle = \sum_{t=0}^{L} |\phi_t\rangle \otimes |t\rangle$. We analyze the contribution from each component of H. If $|T\rangle$ starts with qubits $m+1$ through n clear, $H_{initial}$ does not contribute to $H|T\rangle$. If $|T\rangle$ is a computation of U, that is, $|\phi_t\rangle = U_t |\phi_{t-1}\rangle$ for all t, then from $H_{propagate}$ we get:

$$\begin{aligned} H_j T = &-\frac{1}{2} U_j |\phi_{j-1}\rangle |j\rangle - \frac{1}{2} U_j^\dagger |\phi_j\rangle |j-1\rangle \\ &+ \frac{1}{2} |\phi_j\rangle |j\rangle + \frac{1}{2} |\phi_{j-1}\rangle |j-1\rangle \\ &= -\frac{1}{2} |\phi_j\rangle |j\rangle - \frac{1}{2} |\phi_{j-1}\rangle |j-1\rangle \\ &+ \frac{1}{2} |\phi_j\rangle |j\rangle + \frac{1}{2} |\phi_{j-1}\rangle |j-1\rangle \\ &= 0, \end{aligned}$$

for no contribution from $H_{propagate}$.

Finally, if U accepts with probability at least $1-\epsilon$, only H_{final} contributes a penalty to the sum, for an eigenvalue of at most $\frac{\epsilon}{L+1}$.

The hard part of the proof lies in showing the converse. That if there is no $|\eta\rangle$ which U accepts with high probability, then all eigenvalues of H are large. We refer the interested reader to [**1**] for the proof of this.

Upper bound on BQNP

One consequence of this proof of BQNP-completeness is the following:

Theorem: $BQNP \subseteq P^{\#P}$.

Consider the trace of H^k. This is either at least b^k or at most Na^k. We can make sure that $b^k >> Na^k$, by choosing $k >> n^d log N$. So we just need to estimate $Tr(H^k)$ in $P^{\#P}$.

To see this, write $Tr(H^k) = Tr((\sum_j H_j)^k) = Tr(\sum_{j_1,\ldots j_k} H_{j_1} \cdots H_{j_k}) = \sum_{j_1,\ldots j_k} Tr(H_{j_1} \cdots H_{j_k})$. Each trace in this sum is itself just a sum of exponentially many easy to compute contributions, and thus the entire sum is easily seen to be estimated in $P^{\#P}$.

Kitaev's results may well be the first steps towards a rich new theory of BQNP-completeness. Perhaps the most important open question in this area is to find other examples of natural BQNP-complete problems.

References

[1] Aharonov, D, and Nave, T., **Quantum NP**, manuscript.

[2] Ambainis, A., **Quantum lower bounds by quantum arguments**, *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, 2000.

[3] Arora, S., Impagliazzo, R., and Vazirani, U., **On the Role of the Cook-Levin Theorem in Complexity Theory**, manuscript (1994).

[4] Bennett, C. H., **Logical reversibility of computation**, *IBM J. Res. Develop.*, Vol. 17, 1973, pp. 525-532.

[5] Bernstein, E., **Quantum complexity theory**, PhD dissertation, U.C. Berkeley, 1996.

[6] Blum, M. and Impagliazzo, R., **Generic oracles and oracle classes**, 28th Annual Symposium on Foundations of Computer Science, IEEE computer society press, 1987.

[7] Barenco, A., Bennett, C., Cleve, R., DiVincenzo, D., Margolus, N., Shor, P., Sleator, T., Smolin, J., and Weinfurter, H., **Elementary gates for quantum computation**, Phys. Rev. A **52**, 3457 (1995).

[8] Beals, R., Buhrman, H., Cleve, R., Mosca, M., de Wolf, R., **Quantum Lower Bounds by Polynomials**, *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, 1998.

[9] Bernstein, E. and Vazirani, U., **Quantum complexity theory**, *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, 1993, pp, 11-20 . SIAM J. Computing, 26, pp. 1411-1473 (1997).

[10] Bennett, C., Bernstein, E., Brassard, G., and Vazirani, U., **Strengths and weaknesses of quantum computation**, ISI Torino tech report (1994), SIAM J. Computing, 26, pp. 1510-1523 (1997).

[11] Deutsch, D., **Quantum theory, the Church-Turing principle and the universal quantum computer**, *Proc. R. Soc. Lond.*, Vol. A400, 1985, pp. 97-117.

[12] Deutsch, D., "Quantum computational networks," *Proc. R. Soc. Lond.*, Vol. A425, 1989, pp. 73-90.

[13] Deutsch, D. and Jozsa, R., **Rapid solution of problems by quantum computation**, *Proceedings of the Royal Society*, London, vol. A439, 1992, pp. 553-558.

[14] Feynman, R., **Simulating physics with computers**, *International Journal of Theoretical Physics*, Vol. 21, nos. 6/7, 1982, pp. 467-488.

[15] Grover, L., **Quantum mechanics helps in searching for a needle in a haystack**, Phys. Rev. Letters, 78, pp. 325-328 (1997).

[16] Hales,L. and Hallgren, L., **An Improved Quantum Fourier Sampling Algorithm and Applications**, *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, 2000.

[17] Hales, L. and Hallgren, S., manuscript, 1998.

[18] Kitaev, A., **Quantum NP**, AQIP '99.

[19] Kitaev, A., and Watrous, J., **Parallelization, amplification, and exponential time simulation of quantum interactive proof sy stems**, *Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 608-617, 2000.

[20] Manin, Y., **Computable and uncomputable**, Moscow, Sovetskoye Radio, 1980.

[21] Nisan, N., **CREW PRAM's and decision trees**, STOC 1989, pages 327-335.

[22] Nielson, M., and Chuang, I., **"Quantum Computation and Quantum Information,"** Cambridge University Press, 2000.

[23] Papadimitriou, C., **"Computational Complexity,"** Addison-Wesley Publishing Company.

[24] Simon, D., **On the power of quantum computation**, SIAM J. Computing, 26, pp. 1474-1483 (1997).

[25] Shor, P., **Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer**, SIAM J. Computing, 26, pp. 1484-1509 (1997).
[26] Solovay, B., and A. Yao, manuscript 1997.
[27] Vazirani, U., **On the power of quantum computation**, *Philosophical Transactions of the Royal Society of London, Series A*, 356:1759-1768, August 1998.
[28] Watrous, J., **PSPACE has constant-round quantum interactive proof systems**, *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pp. 112-119, 1999.
[29] Watrous, J., **On quantum and classical space-bounded processes with algebraic transition amplitudes**, quant-ph 9911008.
[30] Yao, A., **Probabilistic Computations: Towards a Unified Measure of Complexity**, *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science*, 1977, pp. 222-227.
[31] Yao, A., **Quantum circuit complexity**, *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, 1993.

University of California, Berkeley, Berkeley, CA 94720
E-mail address: vazirani@cs.berkeley.edu
URL: http://www.cs.berkeley.edu/~vazirani